CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

GUSTAVO FONTANA SUZUKAWA

PROTÓTIPO DE UMA IMPRESSORA BRAILLE PARA USO DOMÉSTICO

Orientadora: MSc. Maria Marony Sousa Farias Nascimento

Brasília

dezembro, 2010

GUSTAVO FONTANA SUZUKAWA

PROTÓTIPO DE UMA IMPRESSORA BRAILLE PARA USO DOMÉSTICO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: MSc. Maria Marony Sousa Farias Nascimento.

Brasília

dezembro, 2010

GUSTAVO FONTANA SUZUKAWA

PROTÓTIPO DE UMA IMPRESSORA BRAILLE PARA USO DOMÉSTICO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: MSc. Maria Marony Sousa Farias Nascimento.

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. Maria Marony, Mestre em Engenharia Elétrica
Orientadora

_____

Prof. Vera Farini, Mestre em Matemática
Instituição

_____

Prof. Francisco Javier, Mestre em Engenharia Elétrica
Instituição

AGRADECIMENTOS

Agradeço primeiramente à minha família, base para a conclusão do meu curso superior. De maneira especial lembro-me de meu pai, Sussumu Ohashi Suzukawa, participante vital deste projeto.

Também agradeço à orientadora Maria Marony, colaborando para solucionar problemas encontrados durante o desenvolvimento do trabalho; assim como todos os outros professores do curso de Engenharia de Computação do UniCEUB, inclusive aqueles que atualmente não se encontram mais na instituição. É importante lembrar as contribuições de José Carlos do Laboratório de Eletrônica, Rafael de Mello, Alcides Rafael, Jean Matheus e Gabriel Santos.

Por fim, declaro meus agradecimentos à Associação Brasiliense de Deficientes Visuais e seus membros, bem como à Associação de Amigos do Deficiente Visual.

“Dentro da cela Braille, a liberdade para quem não vê.”

Edison Ribeiro Lemos

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O presente projeto tem por objetivo desenvolver uma solução integrada para os deficientes visuais quanto ao uso de um computador: um *software* transcritor de textos para o sistema Braille e uma impressora capaz de imprimir tais textos em relevo. O *software* foi programado com linguagem de programação Java e recebe textos em língua portuguesa com o intuito de transcrevê-los e enviar o resultado à impressora. Esta é responsável por receber comandos relativos a tal resultado para imprimir o texto transcrito anteriormente em uma folha de papel. A comunicação entre o *software* e a impressora é realizada por um microcontrolador ATmega2560 presente em uma placa Arduino Mega 2560, o qual recebe o resultado do texto, o interpreta e envia comandos para a impressora. Os testes realizados e os respectivos resultados apontam uma transcrição de textos fiel ao sistema Braille e impressões de boa qualidade, permitindo a leitura do conteúdo existente na folha. A partir de uma visão global percebe-se que o projeto cumpre as suas propostas e assiste aos deficientes visuais adequadamente.

Palavras-chave: Sistema Braille, deficiente visual, transcrição, Arduino, impressora matricial.

ABSTRACT

The current project has the goal of developing a integrated solution for the visually challenged people about the computer use: a text transcriber software for the Braille system and a printer capable of printing such texts in relief. The software was programmed with Java programming language and receives texts in Portuguese language with the goal of transcribe them and send the result to the printer. This is held responsible for receiving relative commands to that result to print the previously transcribed text in a paper. The communication between the software and the printer is made by a ATmega2560 microcontroller present in a Arduino Mega 2560 board, which receives the text result, interprets it and sends commands to the printer. The executed tests and respective results indicate a loyal text transcription to the Braille system and high quality prints, allowing the reading of the existent content in the paper. From a global view it is realized that the project fulfills its proposals and properly assists the visually challenged.

CAPÍTULO 1 – INTRODUÇÃO

1.1 – Apresentação do Problema

Avanços tecnológicos influenciaram diversas áreas do conhecimento humano ao longo das últimas décadas, uma nítida e rápida evolução em muitos segmentos da sociedade. Não raro tais avanços foram direcionados a ramos específicos, seja por uma questão monetária ou para abranger um mercado direcionado. No entanto, algumas dessas áreas sofreram uma mudança menor quando comparadas com outras, sendo este o caso da deficiência visual.

Os deficientes visuais carecem de soluções tecnológicas significativas desde o século XIX, quando Louis Braille impactou o mundo com seu sistema de escrita e leitura em relevo. Até a consolidação da era do conhecimento pouco se havia feito para aqueles que possuem dificuldades para enxergar. Após a tendência dos computadores pessoais muitas soluções informatizadas eclodiram neste campo. *Softwares* e impressoras específicos passaram a ser comercializados, mas atualmente a sua maioria possui um custo elevado para serem adquiridos por um usuário pessoal.

Os *softwares* da área possuem a função de transcrever textos de uma língua específica para um sistema de leitura em relevo, quase sempre o sistema Braille. Consistem de programas de código livre ou fechado, gratuitos ou pagos. Em geral possuem uma boa qualidade.

Tratando-se especificamente das impressoras Braille (como são conhecidas as impressoras para deficientes visuais) a situação é agravada pela ausência de uma indústria nacional. Todas as alternativas existentes são de produção estrangeira e no Brasil são poucos os locais que oferecem manutenção para tais impressoras. A aquisição de uma solução dessas por um usuário doméstico torna-se inviável.

1.2 – Objetivos

O projeto possui o objetivo de desenvolver um protótipo de impressora que imprima textos em Braille na língua portuguesa, a partir de uma impressora matricial existente. Essa é a sua meta básica, mais genérica.

Como objetivo específico pode-se citar o desenvolvimento de uma aplicação que receba textos com caracteres alfanuméricos, transcreva-os para o sistema Braille e se comunique com uma impressora matricial existente adaptada para o projeto. Construir um circuito que atue como interface entre o microcontrolador utilizado e os mecanismos da impressora adaptada também é outra meta. Por fim, inclui-se o ato de validar o uso da impressora adaptada com diversos textos.

1.3 – Justificativa e Importância do Trabalho

Solucionar totalmente o problema das impressoras Braille apresentado anteriormente somente é possível com a formação de uma indústria nacional do ramo. Portanto, minimizar a dificuldade de acesso a impressões em Braille por parte dos deficientes visuais através da redução dos custos dos componentes do projeto é uma forma de amenizar tal problema, voltado ao usuário doméstico. Além disso, a solução desenvolvida pelo projeto expande a inclusão digital dos deficientes visuais, a qual todos têm direito.

Alcançar todos os objetivos específicos servirá como alicerce para o cumprimento do objetivo geral proposto. A programação da aplicação transcritora, a construção do circuito de interface e a validação da impressora, concluídas são etapas essenciais para o correto desenvolvimento do projeto.

A intenção de reduzir custos de uma impressora Braille também é uma preocupação almejada, visto que a quantia monetária desembolsada para a aquisição das alternativas existentes impulsiona a conclusão do projeto.

1.4 – Escopo do Trabalho

O escopo do projeto consiste em disponibilizar um protótipo de impressora Braille para textos em português e uma aplicação que realize a transcrição de tais textos com caracteres para o sistema de leitura "Braille de Seis Pontos". A aplicação, desenvolvida com linguagem de programação Java, é executada na plataforma *Windows* e obedece à Grafia Braille para a Língua Portuguesa (portaria nº 2.678 e NBR 9050/2004) com exceções. A impressora utiliza folhas de papel avulso de tamanho A4 com gramatura de 120 g/m^2 para impressão. Os caracteres abrangidos pelo protótipo são somente aqueles suportados pelo componente Java chamado *JTextArea*. O protótipo disponibilizado utiliza uma impressora matricial existente com funcionalidades modificadas, adquirida de terceiros. A comunicação existente entre a aplicação e a impressora é feita por uma placa Arduino Mega 2560 e a interface (circuito eletrônico) entre esse último e a impressora. As impressões são feitas em somente um lado da folha.

O escopo do projeto não contempla outras plataformas que não sejam *Windows*, outras línguas que não sejam a portuguesa (tanto para textos com caracteres como para textos em Braille), outros sistemas além do Braille de Seis Pontos e tamanhos e gramaturas de papel diferentes dos escolhidos. A aplicação não aborda a Escrita Braille em Contexto Informático (um conjunto específico de caracteres do Sistema Braille). O protótipo não possui todas as funções da impressora matricial original, apenas aquelas necessárias ao desenvolvimento do projeto.

1.5 – Resultados Esperados

Como resultado do projeto desenvolvido espera-se que a impressão elaborada seja possível de se ler naturalmente por uma pessoa que tenha conhecimento sobre o sistema Braille, principalmente por um deficiente visual. Essa validação é extremamente importante porque a solução desenvolvida é voltada para aqueles que só conseguem realizar a leitura através do Braille.

De maneira complementar, o *software* programado deve transcrever textos da língua portuguesa para o sistema Braille de maneira correta. A simbologia apresentada necessita estar de acordo com a Grafia Braille para garantir uma boa compreensão do texto lido.

Por fim, os componentes utilizados da impressora matricial precisam atuar sempre que requisitados. Tal expectativa é justificada pelo fato de que o projeto não cumprirá seu objetivo geral se o *hardware* utilizado não for confiável.

1.6 – Estrutura do Trabalho

Todo o trabalho desenvolvido ao longo do projeto encontra-se dividido em seis capítulos. Eis a sua disposição cardinal:

- Capítulo 1: introduz o tema abordado, bem como sua relevância destacada. Apresenta a justificativa para o desenvolvimento do projeto.
- Capítulo 2: trata do cenário atual das tecnologias existentes na área referida. Aponta soluções aplicadas no mercado.
- Capítulo 3: abrange todo o referencial teórico necessário para a compreensão daquilo que foi desenvolvido.
- Capítulo 4: a solução projetada encontra-se nesse capítulo, assim como a sua explicação detalhada e forma de funcionamento.
- Capítulo 5: todos os testes realizados sobre a solução estão listados e aprofundados aqui. Entre eles estão a opinião de usuários do sistema Braille.
- Capítulo 6: conclui todo o trabalho avaliando o cumprimento dos objetivos propostos.

CAPÍTULO 2 – CENÁRIO ATUAL DAS TECNOLOGIAS BRAILLE

A prática de um indivíduo com alguma necessidade educacional especial frequentar uma instituição de ensino é atual. No entanto, a maneira como isso deve ser feito, as propostas e os objetivos devem ficar claros, e geralmente essas instituições não estão preparadas por diversos motivos (MACHADO, 2009).

Entre os educandos estão os deficientes visuais, aqueles que se encontram entre a baixa visão e a cegueira. A baixa visão é a diminuição da capacidade visual que não impossibilita a leitura de materiais impressos a tinta. A cegueira é a situação a qual leva o indivíduo a necessitar do Sistema Braille para leitura e escrita (MACHADO, 2009).

O Sistema Braille é um sistema de escrita e leitura tátil que permite a representação de letras, números, acentuação, pontuação e símbolos básicos de aritmética, sendo inventado por Louis Braille na França do século XIX (ABREU, 2008). Com o passar dos anos e sua adoção pelo mundo, o método foi adaptado de acordo com o país adotante, inclusive no Brasil com a língua portuguesa.

Em todo o planeta, as maneiras de escrita em papel do sistema Braille são quase sempre as mesmas: punção e reglete, máquina de escrever em Braille e impressora Braille. O punção e a reglete são instrumentos totalmente manuais e por muito tempo foram a única solução de escrita Braille. A máquina de escrever em Braille é um pouco mais sofisticada, consistindo de teclas representando os seis pontos da escrita. Por fim, a impressora Braille é o grande destaque para escrita desse tipo e traz muitas facilidades àqueles que a utilizam.

Todos desejam fazer uso da informática com a sua crescente expansão entre nós, inclusive os deficientes visuais. Um desses usos ocorre através da impressão de textos em Braille, possibilitado pelas já citadas impressoras Braille. No entanto, essas impressoras possuem um alto custo no mercado brasileiro devido à pequena concorrência existente e por serem todas importadas (GRANDI, 2010). O fato dos deficientes visuais constituírem uma parte da sociedade que pouco é lembrada e pouco recebe assistência das demais pessoas também contribui para a redução da variedade de impressoras Braille disponíveis.

Um equipamento desses é ideal quando se precisa imprimir textos diretamente de um computador pessoal, ou seja, sem a necessidade de terceiros, como as gráficas. Ele traz um alto grau de independência ao seu usuário e um grau ainda maior se esse for deficiente visual, satisfazendo um dos principais objetivos daqueles que possuem a visão limitada.

2.1 – Impressoras Braille Existentes

As impressoras Braille podem ser classificadas em interponto (imprimem frente e verso) e não-interponto (imprimem em apenas um lado da folha), possuem interface paralela ou USB, utilizam formulário contínuo ou folhas soltas, e suportam apenas textos ou também gráficos e desenhos em relevo. A seguir estão alguns exemplos (GRANDI, 2010):

A *Juliet Pro-60*, como mostrado na figura 2.1, é uma impressora Braille bem aceita pelos consumidores. Possui tamanho reduzido, é do tipo interponto, imprime folhas soltas e formulário contínuo. Além disso, emite avisos sonoros antes e depois da impressão e é ideal para pequenos trabalhos do dia-a-dia.

Figura 2.1 - Impressora Braille Juliet Pro-60. (Fonte: http://www.brailler.com/juli3.htm)

Outra impressora existente no mercado é a *Basic D*, como mostrado na figura 2.2. Atua com formulário contínuo, imprime até 50 caracteres por segundo e é não-interponto. Seu painel frontal é composto por instruções em tinta e em Braille e possui um sintetizador de voz interativo em português.

Figura 2.2 - Impressora Braille Basic D. (Fonte: http://www.indexbraille.com/Products/Embossers/Basic-D.aspx)

A família de impressoras *Tiger* (figura 2.3) é a melhor opção para a impressão de desenhos, mapas, gráficos e textos em relevo. Seus modelos suportam vários tipos de papel e se destacam no mercado por permitirem a configuração do nível de relevo da impressão, com o objetivo de diferenciar textos e desenhos no papel impresso. A *Tiger Pro* faz parte dessa família e interage bem com as aplicações do sistema operacional *Windows*.

Figura 2.3 - Impressora Braille Tiger Pro. (Fonte: http://www.viewplus.com/products/braille-printers/pro-braille-printer/)

Alternativamente às anteriores, tem-se a Index 4x4 Pro (figura 2.4), capaz de produzir livros. Suporta a impressão interponto, atua com folhas soltas e seu intervalo de manutenção é extenso. Oferece um trabalho de acabamento mais sofisticado devido ao material produzido.

Figura 2.4 - Impressora Braille Index 4x4 Pro. (Fonte: http://www.indexbraille.com/Products/Embossers/4X4-PRO.aspx)

Para se adquirir uma impressora Braille é preciso desembolsar uma quantia considerável, uma barreira para muitos. As impressoras citadas possuem custos elevados para compra e manutenção. Deve-se levar em consideração que muitas entidades que oferecem assistências aos deficientes visuais, potenciais consumidoras, possuem orçamentos limitados, e o alto custo de uma aquisição dessas interfere negativamente na hora de decidir por sua compra.

2.2 – Softwares de Transcrição Existentes

Outro ponto importante na impressão de materiais em Braille é o *software* utilizado para realizar a transcrição dos textos desejados. Existem opções comerciais e outras gratuitas, muitos com uma boa qualidade, mas somente os *softwares* comercializados oferecem suporte (GRANDI, 2010). Até o momento não se conhece um programa desse tipo de código livre com suporte à língua portuguesa, ou seja, uma alternativa aberta com contribuição da comunidade de software livre.

Esse tipo de *software* atua como ponte entre o texto alfanumérico e o correspondente em Braille. Geralmente se comunica com as principais impressoras Braille do mercado para posterior impressão do conteúdo transcrito. Porém, por realizar a transcrição de uma maneira automática e aplicar regras estáticas, muitas vezes seu resultado pode ser insatisfatório para o usuário. É ideal a revisão por um indivíduo conhecedor do sistema Braille daquilo que foi transcrito nos casos mais críticos.

O programa de transcrição de textos conhecido como *Braille Fácil* é gratuito, simples de utilizar e uma alternativa às opções comerciais. É constituído de um editor de textos para digitação ou importação de conteúdo escrito e se comunica diretamente com as impressoras Braille. A figura 2.5 mostra a sua tela de interação com o usuário.

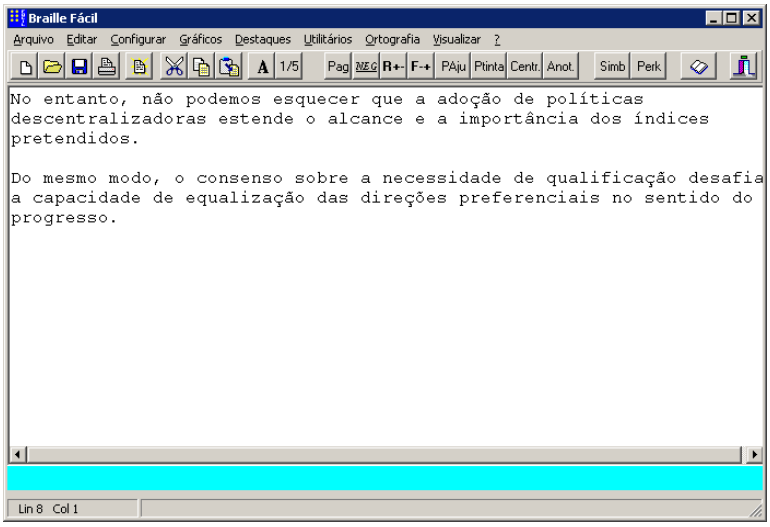

Figura 2.5 - Software de transcrição Braille Fácil. (Fonte: Autor)

Um programa comercial dessa natureza é o *Braille Creator*. Possui a mesma função central do anterior, mas oferece suporte ao consumidor e se comunica com um número maior de impressoras Braille. É executado em ambiente *Windows* e abrange uma gama maior de recursos para edição de textos.

O *Duxbury Braille Translator* é outro *software* comercial com o mesmo intuito. Suporta diversas línguas (inclusive a portuguesa), consegue importar arquivos de editores de texto consagrados, suporta caracteres matemáticos e está há mais de duas décadas no mercado. Sua interface se encontra na figura 2.6.

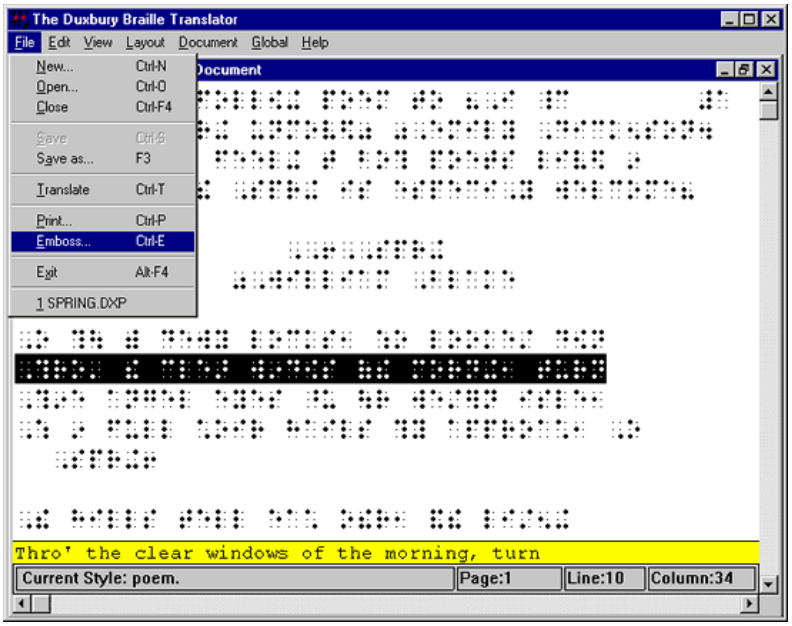

Figura 2.6 - Software de transcrição Duxbury Braille Translator. (Fonte: http://www.specialneedscomputers.ca/index.php?l=product_detail&p=355)

Atualmente a aquisição de uma impressora Braille quase sempre resulta na obtenção de um *software* transcritor, visto que são ferramentas complementares. Em outras palavras, é necessário adquirir ambos separadamente para se obter um conjunto completo de transcrição e impressão de textos Braille. Isso pode acarretar em um problema de compatibilidades entre ambos, pois os *softwares* nem sempre suportam todas as impressoras existentes no mercado.

Visto isso, o projeto desenvolvido procura oferecer uma solução única: um programa transcritor que se comunique com a respectiva impressora. Apesar de a compatibilidade existir apenas com essa impressora, a solução é válida e expande a participação dos deficientes visuais na informática.

CAPÍTULO 3 – REFERENCIAL TECNOLÓGICO

O projeto desenvolvido utiliza métodos e ferramentas diversas para alcançar seus objetivos. O sistema de leitura e escrita Braille, linguagens de programação, o barramento USB, um microcontrolador e uma impressora matricial compõem o projeto. Existem razões para a escolha de cada um desses módulos, bem como há maneiras corretas de aplicá-los a fim de atingir seu correto funcionamento.

3.1 – Sistema de Leitura e Escrita Braille

Os deficientes visuais, devido à sua limitação física, não são capazes de compartilhar da leitura e escrita em tinta com as pessoas que enxergam. Esse fato, por si só, geraria uma exclusão social, onde quem não possuísse o sentido da visão seria privado de se comunicar com os outros de uma maneira que não fosse através de sons.

No entanto, existe uma solução para tal grave problema: o chamado Sistema Braille. Esse é um sistema de leitura e escrita tátil baseado em pontos que permite a representação de letras, números, acentuação, pontuação e símbolos de aritmética, voltado para as pessoas cegas (ABREU, 2008). Foi inventado pelo francês Louis Braille em 1824 no Instituto Real dos Jovens Cegos, em Paris. No ano de 1837 foi concebida sua versão final, adotada até hoje ao redor do planeta com as adaptações necessárias de cada localidade.

Atualmente no Brasil o sistema Braille é chamado de Grafia Braille para a Língua Portuguesa, sendo essa regulamentada pela portaria nº 2.678 de 24 de setembro de 2002 através da Secretaria de Educação Especial do Ministério da Educação. Este documento se encontra em sua segunda edição, datada do ano de 2006. Tal portaria aprova o projeto da Grafia Braille para a Língua Portuguesa e recomenda o seu uso em todo o território nacional (BRASIL, 2006).

O sistema Braille é o centro do projeto desenvolvido. Ele se encontra em várias etapas de execução, onde a todo momento é referenciado. Na aplicação transcritora é utilizado para realizar-se a transcrição do texto inserido, no microcontrolador cada cela Braille é

representada por *bits* e a impressora se encarrega de imprimir o resultado final em relevo no papel.

3.1.1 – Estrutura

O sistema Braille é constituído por 64 sinais formados por pontos a partir do conjunto matricial ⠿ (123456). É conhecido como Sistema Braille de Seis Pontos. Tal conjunto de seis pontos é chamado de sinal fundamental. O espaço ocupado por esse conjunto é denominado cela braille. Os pontos são numerados de cima para baixo e da esquerda para a direita. Abaixo segue a relação entre os pontos e sua numeração.

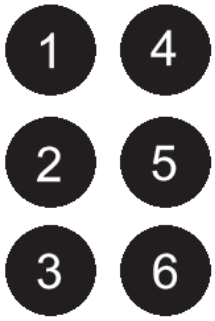

Figura 3.1 – Numeração dos pontos em uma cela braille. (Fonte: BRASIL, 2006)

As palavras são formadas por sinais escritos consecutivamente, como em um texto à tinta. Tais sinais recebem designações diferentes de acordo com o espaço que ocupam. Os sinais simples são aqueles que ocupam somente uma cela, enquanto os sinais compostos são obtidos através da combinação de dois ou mais sinais simples (BRASIL, 2006).

Cada sinal simples caracteriza-se pela presença ou ausência dos seis pontos que formam cada cela braille. Por exemplo, a letra “a” é representada pela presença do ponto número 1 e ausência dos demais pontos.

3.1.2 – Valor dos sinais

Os sinais empregados na escrita dos textos em língua portuguesa são muitos. Logo, há uma complexidade considerável quanto à variedade existente de combinações de pontos. Abaixo segue o alfabeto da língua portuguesa no sistema Braille. Um ponto menor significa ausência daquele ponto, enquanto que um ponto maior significa presença daquele ponto na cela braille.

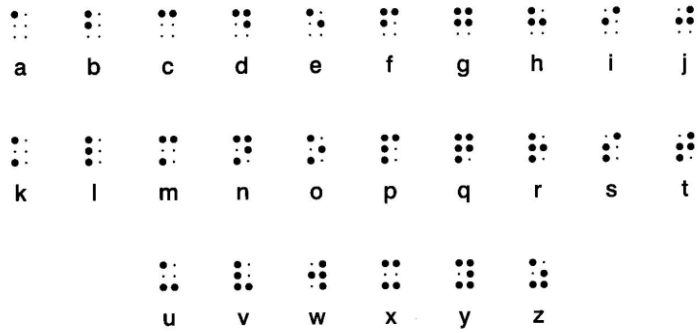

Figura 3.2 – Alfabeto da língua portuguesa em Braille. (Fonte: ABREU, 2008)

Com a ausência do caractere “c com cedilha” do alfabeto acima, ele é representado pelo sinal ⠯ (12346).

As letras com diacríticos em Braille foram adaptadas à nossa língua, ou seja, apenas aqueles que constam em nosso idioma fazem parte do sistema. Os que não integram o sistema foram excluídos da Grafia Braille para a Língua Portuguesa.

Vogais	a	⠁	e	⠑	i	⠊	o	⠕	u	⠥
Acento agudo	á	⠷	é	⠿	í	⠌	ó	⠬	ú	⠾
Acento grave	à	⠮	–	–	–	–	–	–	–	–
Acento circunflexo	â	⠡	ê	⠣	–	–	ô	⠹	–	–
Til	ã	⠜	–	–	–	–	õ	⠒	–	–
Trema	–	–	–	–	–	–	–	–	ü	⠳

Figura 3.3 – Letras com diacríticos em Braille. (Fonte: BRASIL, 2006)

Os sinais de pontuação e sinais acessórios também integram o sistema Braille.

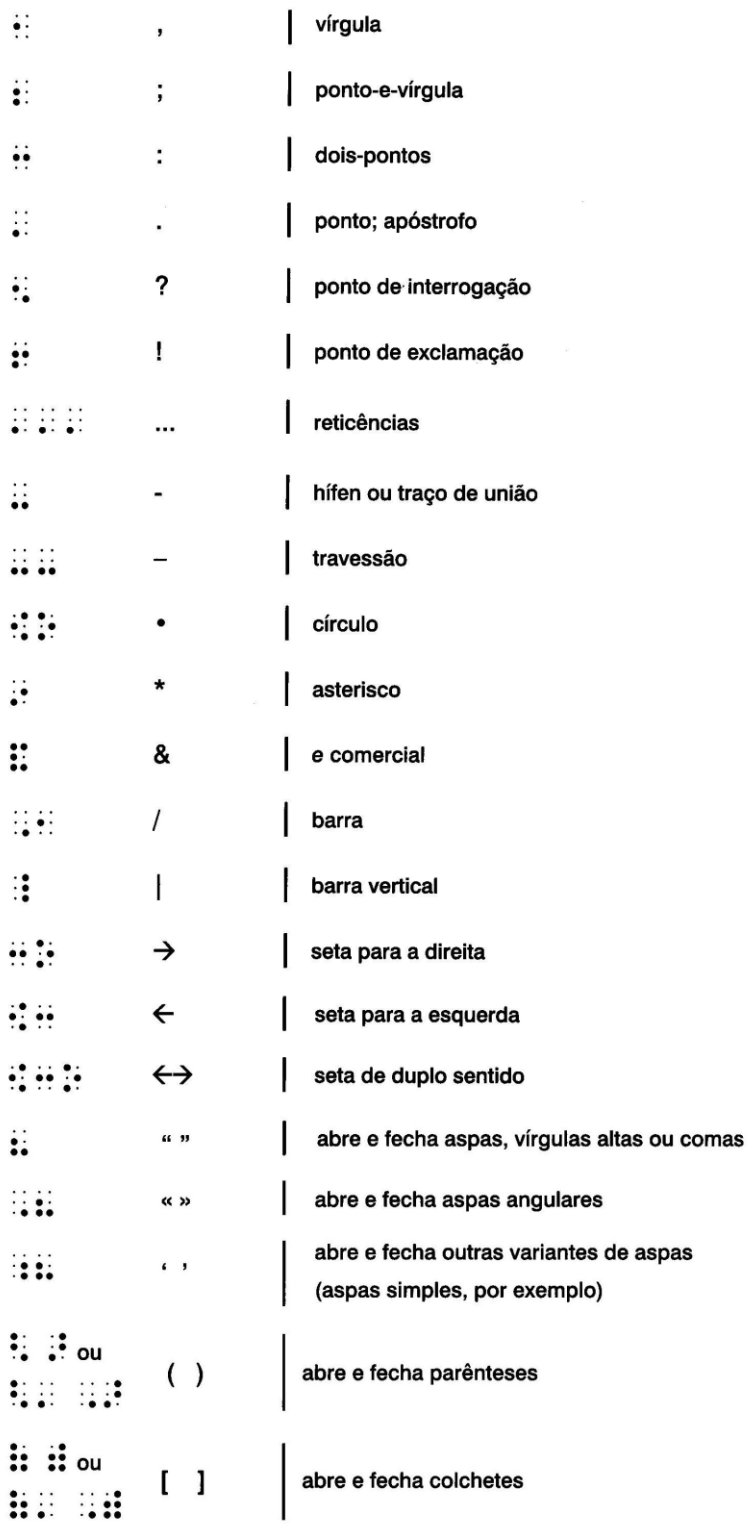

Figura 3.4 – Sinais de pontuação e sinais acessórios em Braille. (Fonte: ABREU, 2008)

Tem-se ainda os sinais usados com números e aqueles exclusivos da escrita Braille.

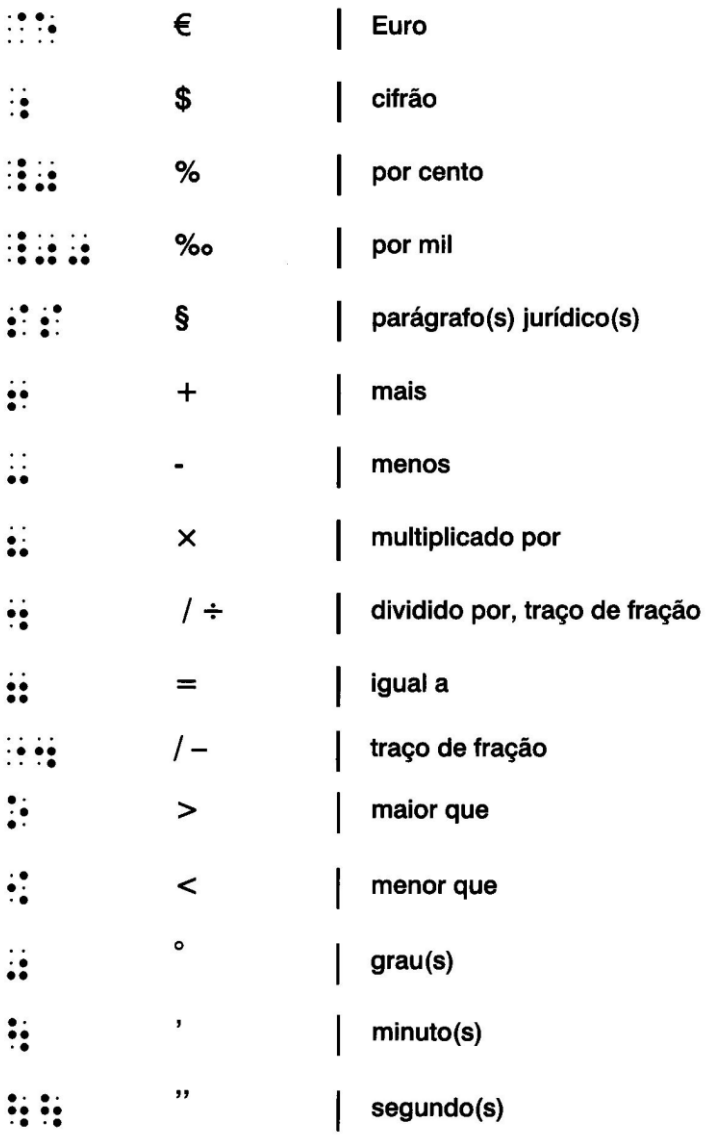

Figura 3.5 – Sinais usados com números em Braille. (Fonte: ABREU, 2008)

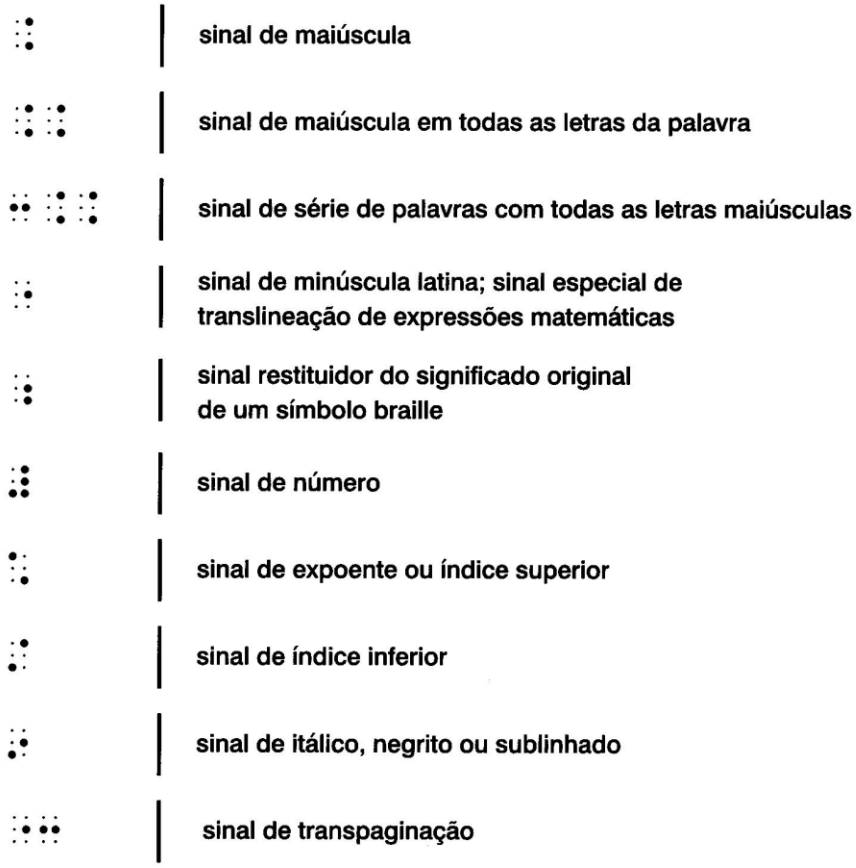

Figura 3.6 – Sinais exclusivos da escrita Braille. (Fonte: ABREU, 2008)

A representação de números é feita por sinais compostos. Ao se colocar o sinal ⠼ (3456) precedendo uma das letras de “a” até ”j”, forma-se um número que varia de 0 até 9. Se o número é formado por mais de um algarismo, simplesmente adiciona-se outro sinal ao seu fim. O sinal indicativo de número só aparece uma vez para cada número. Alguns exemplos se encontram abaixo.

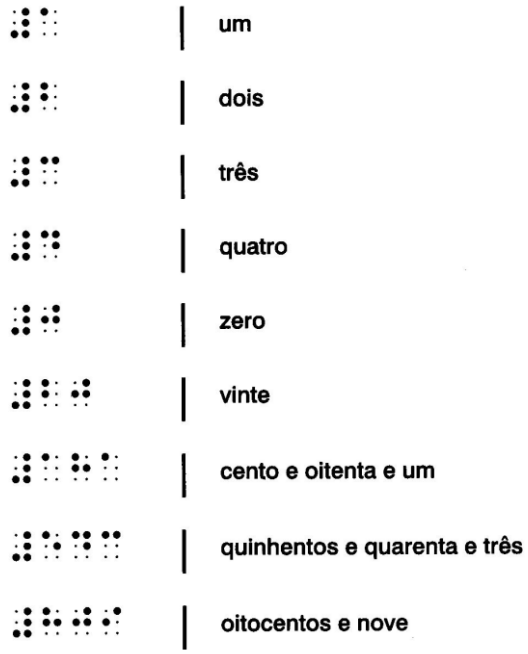

Figura 3.7 – Exemplos de números em Braille. (Fonte: ABREU, 2008)

3.1.3 – Transcrição

Transcrição no âmbito Braille é o ato de representar em símbolos em relevo algum caractere que esteja em tinta, ou seja, plano. Quando se possui um texto em alguma língua e deseja-se "traduzi-lo" para o sistema Braille, ocorre a transcrição (ABREU, 2008).

A transcrição é um processo subjetivo, o que acarreta uma dificuldade por parte de quem o faz. Muitos símbolos do sistema Braille são relativos ou representam mais de um caractere em tinta. Como agravante, é uma etapa a qual demanda um período de tempo considerável.

Uma solução parcial são os programas de computador. Os *softwares* de transcrição, também chamados de *softwares* transcritores, realizam essa função. Eles auxiliam bastante quando há uma situação de grande volume de caracteres a serem transcritos, poupando tempo. Porém, pelo fato da transcrição ser um processo subjetivo, em muitas ocasiões os resultados de tais programas não são satisfatórios ou poderiam ser passíveis de melhora. Ainda assim, sem dúvida esses *softwares* representam uma grande evolução na transcrição de textos em tinta para o sistema Braille.

3.1.4 – Produção e tecnologia

Por muito tempo a escrita em Braille somente foi possível através de ferramentas totalmente manuais, como o reglete e a punção, e mecânicas, caso da máquina de escrever em Braille. Atualmente existem impressoras Braille de pequeno e grande portes para tal finalidade.

Seja qual for a ferramenta utilizada, existem medidas para a cela braille que devem ser seguidas no momento da produção de textos em papel para garantir-se uma padronização. Todas elas são descritas pela Norma Brasileira 9050 (ABNT, 2004).

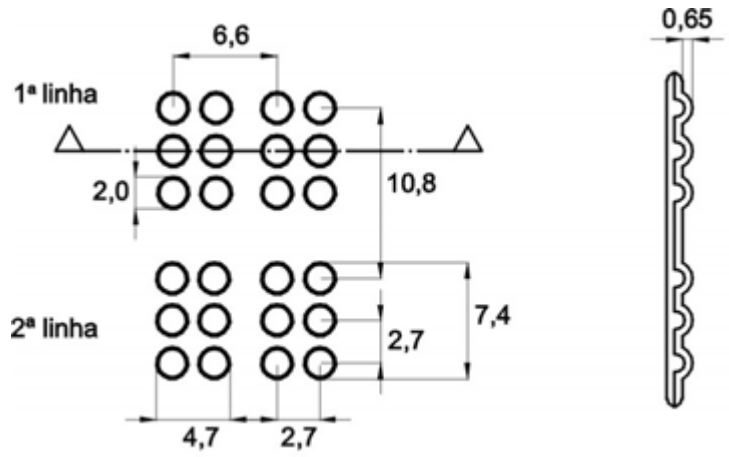

Figura 3.8 – Medidas da cela braille em milímetros, com vista superior à esquerda e vista de corte à direita. (Fonte: ABNT, 2004)

3.2 – Linguagem de Programação

Como dito anteriormente, o texto desejado será transcrito para o sistema Braille através de uma aplicação transcritora. Essa aplicação desenvolvida no projeto requer domínios de conhecimentos na área de linguagens de programação para a compreensão do seu funcionamento técnico.

Por uma linguagem de programação um computador é capaz de executar tarefas. Ele interpreta os comandos existentes nos algoritmos dessa linguagem e executa ações de acordo com o que foi programado anteriormente.

As linguagens de programação se subdividem em três tipos: linguagens de máquina, linguagens de montagem (ou linguagens de baixo nível) e linguagens de alto nível (DEITEL, 2006). As linguagens de máquina são constituídas somente de *bits*, ou seja, apenas sequências de 0s e 1s. As linguagens de montagem ou baixo nível utilizam abreviações de palavras em inglês para representar operações elementares e podem ser traduzidas para linguagens de máquina. As linguagens de alto nível simplificam a maneira de se programar e aceleram esse processo com comandos que agrupam operações e executam tarefas substanciais. Podem ser traduzidas para linguagens de máquina. As linguagens de programação Java e Processing são exemplos de linguagem de alto nível.

3.2.1 – Algoritmo

As linguagens de programação são formadas por algoritmos que devem ser coerentes e válidos, de maneira a apresentarem uma boa qualidade. Um algoritmo é uma sequência de passos que visam a atingir um objetivo bem definido (FORBELLONE, 2005).

Esta sequência de passos necessita de uma ordem, uma lógica. Devem ser especificadas ações claras a partir de um estado inicial, com um período de tempo finito, de forma a produzirem resultados previsíveis.

Um algoritmo, uma vez construído, pode ser traduzido para qualquer linguagem de programação e ser agregado a qualquer funcionalidade de determinado ambiente (FORBELLONE, 2005). Esse processo é chamado de codificação e no caso do projeto desenvolvido os algoritmos elaborados são codificados em linguagens de programação Java e Processing.

3.2.2 – Linguagem de programação Java

Esta linguagem foi escolhida para o projeto por ser de prévio conhecimento do autor e por possuir um suporte muito bom, disponível na comunidade *Web*. É bastante difundida, além de permitir a criação de interfaces gráficas e a comunicação com dispositivos pelas portas serial e USB do computador.

A linguagem de programação Java, ou simplesmente Java, é classificada como orientada a objeto, ou seja, é modular e tem os objetos como centro. Um objeto pode ser entendido assim como na vida real, por exemplo um carro, um telefone ou uma casa. Cada objeto possui uma classe que contém atributos e métodos que o caracterizam (DEITEL, 2006).

O Java tem crescido bastante nos últimos anos, seguindo a tendência de adoção de linguagens de programação orientadas a objeto pelo meio corporativo. Além disso, é considerada por muitos profissionais da área fácil de se programar, já está enraizada na comunidade *Web* e periodicamente recebe atualizações na forma de versões liberadas.

No projeto a linguagem Java é utilizada na construção das classes de objetos do programa transcritor. A recepção do texto, o processo de transcrição e o envio do resultado para o microcontrolador são etapas executadas pelo programa e implementadas pelo Java.

3.2.3 – Linguagem de programação Processing

O Processing é a linguagem utilizada pela IDE (*Integrated Development Environment*) Arduino Alpha para programação. Dessa maneira, foi escolhida para a programação do código presente no microcontrolador.

A linguagem de programação Processing é voltada para trabalhar com gráficos, projetada para *designers* e estudantes de arte e objetiva a simplicidade no momento de se programar. Muitas instruções possuem apenas uma linha, o que resulta em flexibilidade para o código desenvolvido, englobando áreas como geometria, sons, vídeos, renderização, dados, redes, imagens, hardware etc (REAS, 2010).

Como já dito, o código a ser executado pelo microcontrolador será programado em linguagem Processing e não em uma linguagem de baixo nível. Isso facilita o desenvolvimento do código por parte do programador, no caso o autor, e garante uma melhora na manutenção desse código.

3.3 – Interface USB

A aplicação presente no computador que transcreve os textos para o sistema Braille deve se comunicar com o microcontrolador por algum meio físico. Para tal tarefa a interface USB é ideal por motivos claros.

A interface USB foi escolhida para a comunicação entre o computador e o microcontrolador devido à sua configuração simples e à existência de portas USB no computador utilizado e ausência de uma porta serial no mesmo. Além disso, a placa Arduino Mega 2560, onde se encontra o microcontrolador ATmega2560, possui somente uma porta USB como forma de comunicação externa de dados.

USB (*Universal Serial Bus*) é uma interface de grande sucesso para computadores pessoais. É capaz de conectar teclados, *mouses*, controladores de jogos, *scanners*, câmeras, impressoras, *drives* e outros dispositivos a um computador.

A interface USB é fácil de usar, possui múltiplas velocidades que variam de 1.5 Mbps à 5 Gbps, é confiável devido aos protocolos utilizados, possui baixo custo, economiza energia e é suportada pelos maiores sistemas operacionais. No entanto, a distância entre o computador e o dispositivo conectado é pequena, não existe comunicação P2P (*Peer-to-Peer*) entre seus dispositivos e computadores considerados legados não possuem portas USB (AXELSON, 2009).

No projeto um cabo conecta o computador à placa Arduino através da interface USB. As taxas de transmissão de dados são configuradas pela aplicação presente no computador e pelo código existente no microcontrolador.

3.4 – Microcontrolador

O microcontrolador cumpre um papel importante no projeto: interpretar comandos da aplicação e interagir com a impressora de forma a imprimir textos em relevo.

Um microcontrolador pode ser entendido como um microprocessador e seus periféricos reunidos em um só *chip* (NICOLOSI, 2007). Internamente à pastilha do microcontrolador existem a CPU (*Central Processing Unit*), a ROM (*Read Only Memory*), a RAM (*Random Access Memory*) e os *timers* como componentes principais, podendo haver outros. Em outras palavras, a arquitetura interna do microcontrolador é diferente daquela do microprocessador, tanto em componentes como em conjunto de instruções.

3.4.1 – Microcontrolador ATmega2560

O ATmega2560 é um microcontrolador de 8 *bits* desenvolvido pela empresa Atmel (figura 3.9). Possui três memórias diferentes: uma *flash* com 256 kB para armazenamento de códigos, uma volátil *SRAM* de 8 kB e outra *EEPROM* de 4 kB. Sua arquitetura é *RISC*, possui um conjunto de 135 instruções e trabalha a uma frequência de 16 MHz. Também é composto por 54 pinos digitais para entrada e saída e 16 pinos analógicos para entrada (ATMEL, 2010). Abaixo segue a configuração do pinos do ATmega2560.

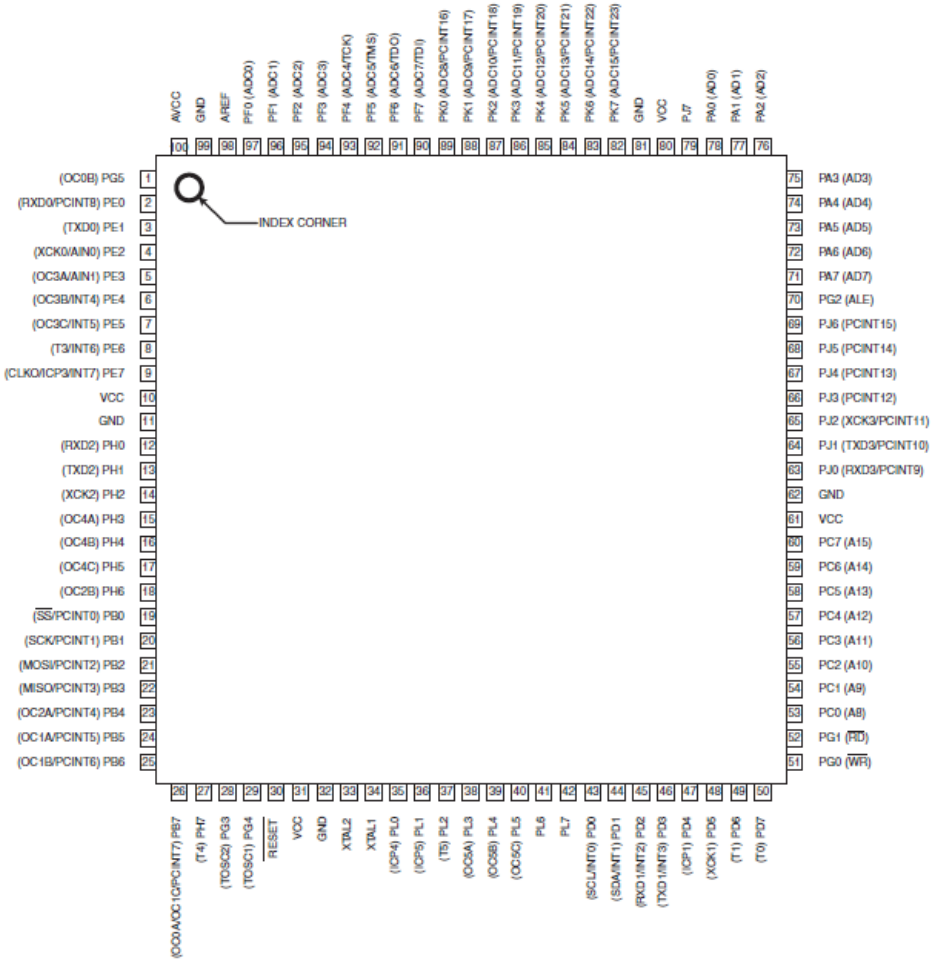

Figura 3.9 – Configuração dos pinos do microcontrolador ATmega2560. (Fonte: ATMEL, 2010)

3.4.2 – Placa Arduino Mega 2560

A escolha dessa placa é devido principalmente ao seu tamanho reduzido e à presença de uma porta USB para comunicação. De uma forma geral ela atende as necessidades do projeto.

A Arduino Mega 2560 é uma placa de entrada e saída (E/S) integrante de uma plataforma de código aberto. Possui o microcontrolador ATmega2560 como base. A plataforma da placa também oferece uma IDE para o desenvolvimento de códigos em linguagem Processing (BANZI, 2009). Para a comunicação com o computador oferece uma porta USB, de onde retira energia para seu funcionamento de forma paralela.

A placa Arduino Mega 2560, como mostra a figura 3.10, é a ponte entre a aplicação transcritora e a interface de ligação com a impressora. Recebe comandos do programa no computador, interpreta-os e executa as devidas ações.

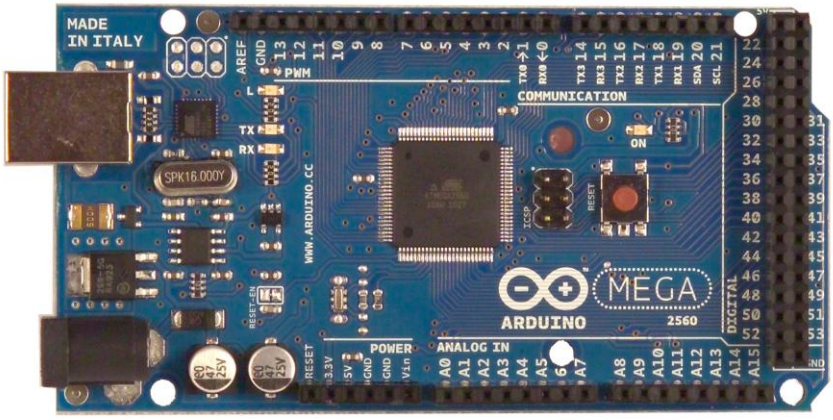

Figura 3.10 – Placa Arduino Mega 2560. (Fonte: http://arduino.cc/en/Main/ArduinoBoardMega2560)

3.5 – Impressora Matricial

A última etapa do projeto justifica todas as outras porque apresenta o resultado esperado. Em uma folha de papel textos são representados pelo sistema Braille graças à impressora matricial do projeto.

A impressora é um clássico periférico de saída de um computador, pelo qual informações são expressas em símbolos em um meio externo, geralmente o papel. As impressoras matriciais fazem parte da categoria de impressoras de impacto. Em sua maioria o mecanismo de impressão desses dispositivos denominados matriciais é composto por um conjunto de agulhas que se projeta contra uma fita contendo tinta, a qual entra em contato com o papel e imprime o símbolo desejado (MONTEIRO, 2002). Os caracteres são impressos sequencialmente e em uma alta velocidade levando-se em conta a sua natureza predominantemente mecânica.

O projeto inclui uma impressora matricial usada devido ao seu baixo custo de aquisição e simplicidade de funcionamento quando comparado a outras tecnologias.

3.5.1 – Impressora Epson LQ-500

A impressora em questão foi escolhida devido à sua facilidade de obtenção junto ao local de trabalho do autor. É um artigo usado e que não era utilizado pela empresa que a possuía, além de satisfazer os requisitos do projeto. A figura 3.11 mostra um exemplar desse modelo.

A Epson LQ-500 é uma impressora matricial da década de 1980 e imprime os caracteres por impacto, utilizando para isso 24 pinos (ou agulhas) (EPSON, 1988). Muitas de suas características são consideradas ultrapassadas, mas ela atende ao propósito do projeto satisfatoriamente. Os seus componentes principais que mais estão ligados ao que se propõe fazer são a cabeça de impressão, o rolo e os roletes de alimentação do papel.

O texto transcrito pela aplicação será impresso pela impressora matricial adquirida, representando a saída, o resultado final. O papel impresso apresentará símbolos Braille em relevo.

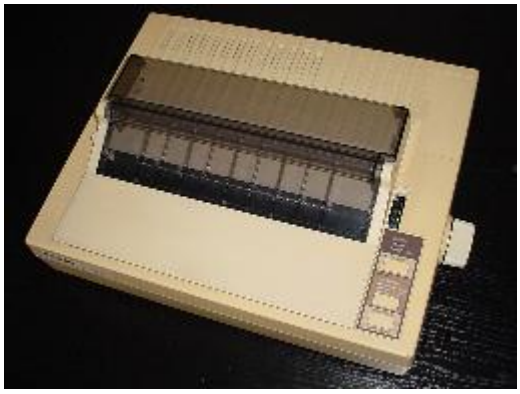

Figura 3.11 – Impressora matricial Epson LQ-500. (Fonte: http://sales.jack.ch/)

3.6 – Motor de passo

Um motor de passo é um dispositivo eletro-mecânico capaz de converter pulsos elétricos em movimentos, obedecendo variações angulares. Tais ângulos são definidos de acordo com o motor utilizado e são denominados passos. Para gerar o movimento de rotação é necessário aplicar impulsos elétricos nos seus terminais em uma sequência correta. Uma sequência implica na rotação do motor em um sentido, enquanto que a mesma sequência invertida rotaciona no sentido contrário. Quanto maior a frequência dos impulsos aplicada, mais rápido o motor é rotacionado (BRITES, 2008).

Por proporcionar movimentos angulares precisos, o motor de passo é amplamente empregado. Seu propósito não é fornecer altas velocidades ou um poderoso torque, mas sim precisão em movimentos e respectivos ângulos de rotação. Dispositivos como impressoras, *scanners*, câmeras e brinquedos fazem uso desse tipo de motor.

Dois motores de passo unipolar de passo inteiro são utilizados no projeto. Em outras palavras, cada uma de suas quatro bobinas é ativada separadamente de cada vez, utilizando sempre somente um pólo e garantindo um passo inteiro na rotação de seu eixo. A figura 3.12 ilustra um exemplo de um motor de passo.

Figura 3.12 – Exemplo de um motor de passo. (Fonte: http://www.cw-motor.com/en/ProductShow.asp?ArticleID=1)

O projeto utiliza todos os conhecimentos expostos anteriormente para atingir seus objetivos. Fica claro a sua subdivisão em módulos evidentes, como a aplicação transcritora, o

microcontrolador e a impressora. A união desses componentes torna possível a proposta do projeto e possibilita a obtenção do resultado esperado.

CAPÍTULO 4 – MODELO PROPOSTO

4.1 – Apresentação Geral

A solução desenvolvida é composta por módulos que cumprem uma função específica e se integram, cada um com um produto diferente, um resultado parcial. Porém, todos são imprescindíveis para o correto funcionamento da solução.

A proposta do projeto é transcrever para o sistema Braille e imprimir um texto qualquer, sempre de acordo com as condições expostas anteriormente. Para tal são utilizados dispositivos de hardware e linguagens de programação, cumprindo papéis distintos que se complementam no final da impressão em Braille. A solução em si está definida em alguns passos.

Primeiramente, através de uma interface do programa transcritor o usuário insere o texto que deve ser transcrito e impresso. Ordena a atuação da lógica do programa, que por sua vez transcreve o texto para o sistema Braille e envia o resultado à placa Arduino. Em seguida o microcontrolador ATmega2560 presente na placa, já de posse do resultado, comanda a execução de tarefas através de suas portas. Elas estão conectadas aos motores que acionam os componentes de uma impressora matricial, como a cabeça de impressão (adaptada com agulhas específicas) e os rolos. Esses componentes atuam de acordo com o programado e imprimem o resultado recebido em uma folha de papel de tamanho A4 e gramatura de 120 g/m^2.

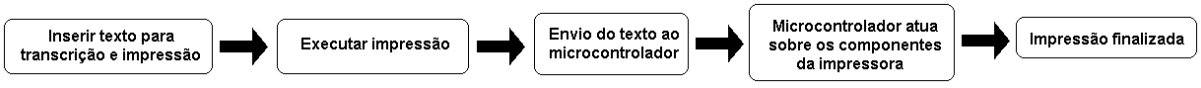

Figura 4.1 – Sequência das etapas da solução. (Fonte: Autor)

Um dos componentes do projeto é um programa denominado “Impressora Braille”, programado em linguagem Java, o qual é executado em um computador (sistema operacional

Windows) com uma conexão USB. Através dessa conexão o microcontrolador se comunica com o computador e também controla uma interface com a impressora matricial. A impressora imprime em relevo os caracteres do texto inseridos na aplicação inicial. Na figura 4.2 segue uma representação da solução desenvolvida.

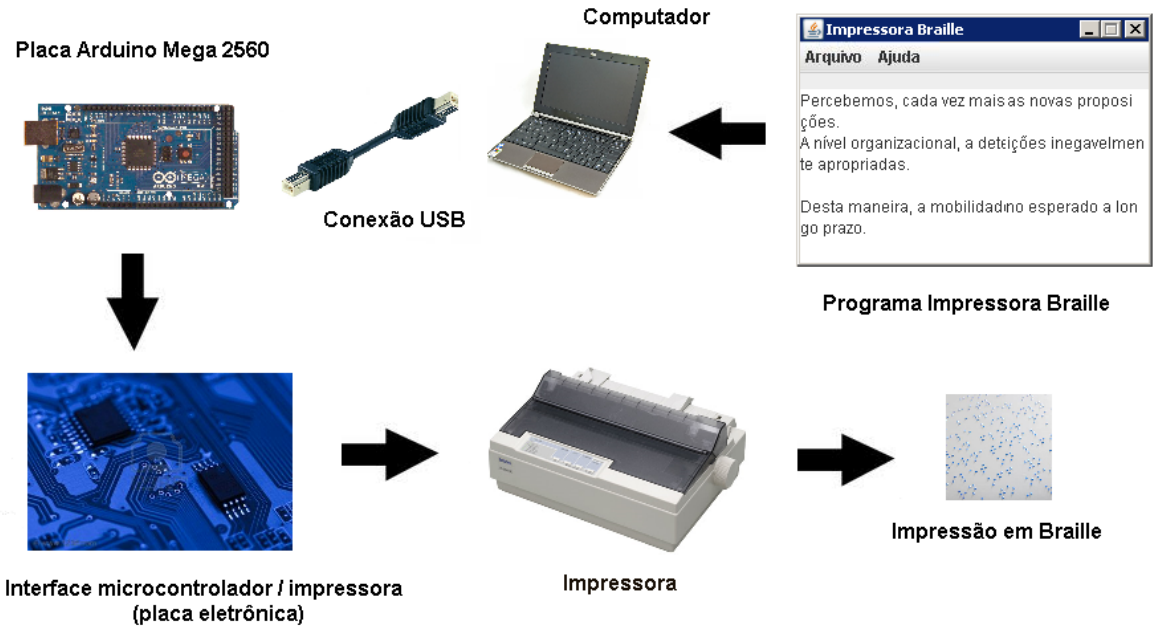

Figura 4.2 – Esquemático da solução desenvolvida. (Fonte: Autor)

4.2 – Software

O *software* do projeto é composto pelo programa "Impressora Braille" e pelo código contido no microcontrolador (Apêndice B). Eles se comunicam e compõem uma vital etapa na impressão dos textos em Braille.

O programa referido é todo programado em linguagem Java e composto por diversas classes diferentes. Já o código do microcontrolador está em uma linguagem chamada *Processing* (REAS, 2010), específica para a sua *IDE* (*Arduino Alpha*) e baseada em linguagem C.

A classes Java são: Impressora, SerialTest, EstruturaAgulhas, TraducaoBraille, ExcecaoCaractereDesconhecido e ExcecaoImpressaoCancelada. Todas elas, com exceção da segunda, foram programadas pelo autor.

A classe Impressora é responsável pela interface gráfica do programa e por coordenar os processos de transcrição e impressão. A classe SerialTest trata da comunicação com a placa Arduino, tanto do envio como do recebimento de dados, e foi baseada em uma classe semelhante disponível no site oficial do Arduino (ARDUINO PLAYGROUND, 2010) que se encontra anexada a este trabalho (anexo A). EstruturaAgulhas é uma classe simples que contém os atributos e métodos necessários para simular a estrutura de uma célula Braille. TraducaoBraille é a classe responsável por transcrever o texto para Braille e retornar o resultado à classe Impressora. As outras duas classes são exceções específicas para um cancelamento de impressão e caracteres desconhecidos no processo de transcrição. Todas as classes desenvolvidas no projeto estão localizadas no apêndice A do trabalho, enquanto que o código presente no microcontrolador encontra-se no apêndice B.

Nas situações possíveis as classes possuem métodos *getters* e *setters*, ou seja, métodos públicos que interagem (retornam ou alteram seu valores) com atributos privados da mesma classe, muito utilizados por objetos de outras classes.

Como observação pode-se citar que o correto funcionamento do projeto desenvolvido implica que o programa transcritor deve estar sendo executado no computador utilizado enquanto o processo de impressão não for completado.

4.2.1 – Requisitos da aplicação

4.2.1.1 – Requisitos funcionais

Os requisitos funcionais do sistema listam as possibilidades de interação com o usuário. São eles:

- O sistema deve aceitar o usuário inserir um texto em português para transcrição;
- O sistema deve transcrever o texto inserido e imprimi-lo ao comando do usuário;

- O sistema deve cancelar a impressão corrente ao comando do usuário;
- O sistema deve enviar ao microcontrolador o texto transcrito.

Os atores do sistema são o usuário e o microcontrolador. O primeiro está envolvido com os três primeiros requisitos funcionais, enquanto que o segundo se encarrega do último requisito. Abaixo seguem os casos de uso do sistema com seus respectivos atores.

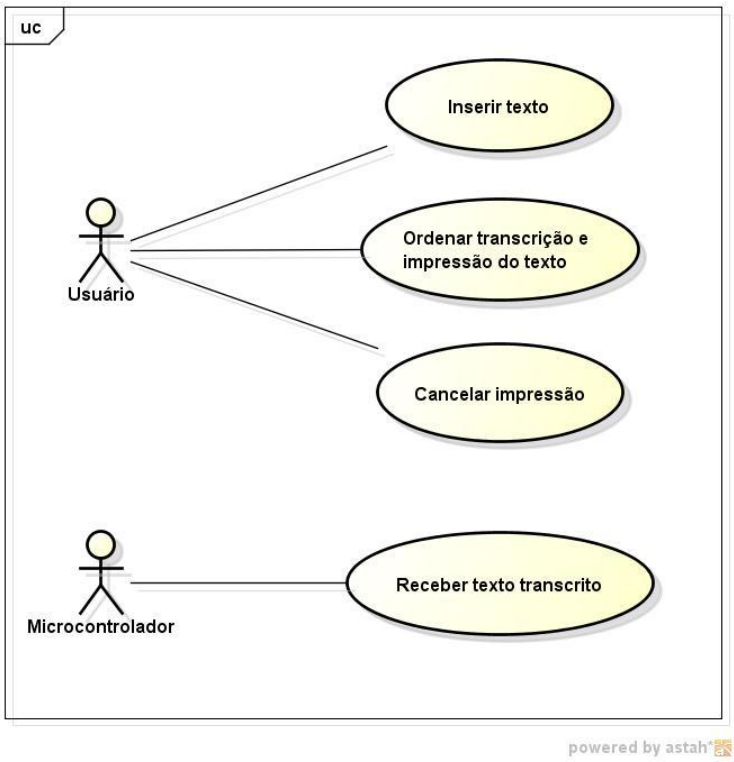

Figura 4.3 – Diagrama de casos de uso. (Fonte: Autor)

4.2.1.2 – Requisitos não funcionais

Existem alguns requisitos não funcionais que se aplicam ao programa desenvolvido, principalmente para auxiliar o modo correto de usá-lo. São requisitos de usabilidade, suportabilidade e performance.

Os requisitos de usabilidade são:

- Interface simples para facilitar o uso do programa;

- Menus de ajuda para orientar o usuário;
- Botões nítidos de impressão e cancelamento de impressão.

Os requisitos de suportabilidade:

- O sistema deve executar na plataforma Windows;
- O sistema deve executar em versões do JDK acima de 1.6;
- O sistema deve suportar conexões USB 1.1 e 2.0;
- O sistema deve suportar textos inseridos em língua portuguesa.

Já os requisitos de performance são:

- O sistema deve terminar o processo de transcrição em até 5 segundos após o clique no botão de "Imprimir";
- O sistema deve iniciar a impressão em até 5 segundos após o término da transcrição.

4.2.2 – Modelo funcional

O programa possui apenas o modelo funcional, uma vez que não armazena dados para posteriores cálculos. Para representar tal modelo é apresentado na figura 4.4 o diagrama de atividades da aplicação contendo os passos a serem executados para a produção da impressão desejada.

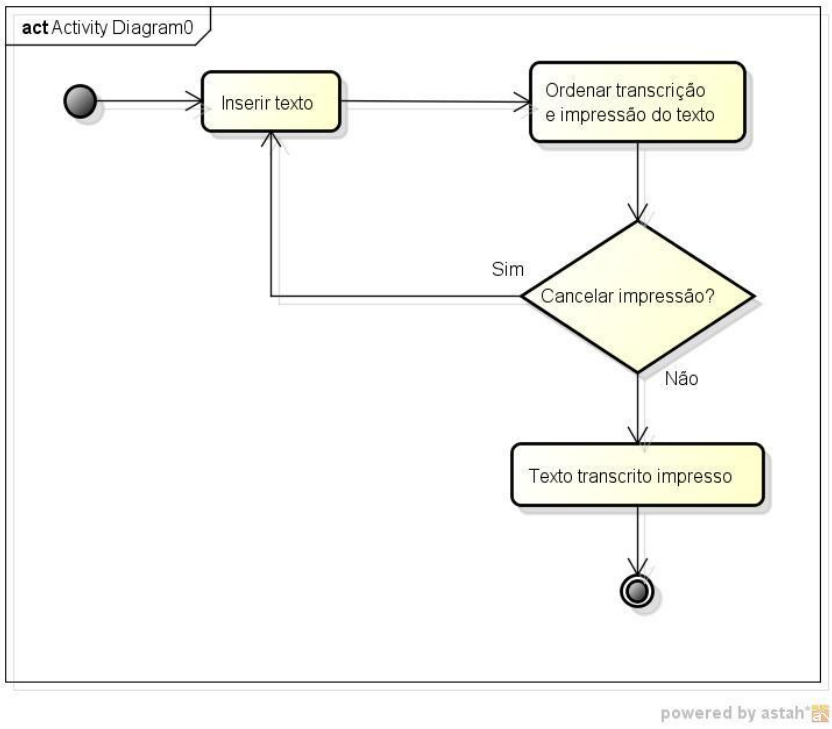

Figura 4.4 – Diagrama de atividades. (Fonte: Autor)

Tanto o diagrama de casos de uso quanto o diagrama de atividades, ambos referentes à aplicação, foram construídos utilizando-se o *software* Astah Community (antigo JUDE).

4.2.3 – Módulo de interface com o usuário

O usuário que deseja imprimir seus textos interage com uma interface construída para o projeto, por meio da qual insere os textos e acompanha a situação de sua impressão. Os caracteres que formam o conteúdo a ser impresso devem estar restritos aos permitidos pelo projeto, ou seja, os listados em seção anterior. A interface está presente na classe Impressora.

A interface citada apresenta uma tela de programa ao usuário com algumas opções pré-determinadas. Seus comandos são simples e possuem apenas as funções essenciais ao funcionamento do projeto. São apresentados uma área para inserção de textos, um botão “Imprimir”, um botão “Cancelar” e quatro itens de menu: “Novo”, “Sair”, “Ajuda do Impressora Braille” e “Sobre”. O pacote de classes do Java conhecido como *Swing* (DEITEL, 2006) permite a construção de interfaces gráficas para aplicações e é utilizado pelo programa.

A área central do programa recebe os textos e é formada por um componente Java denominado *JTextArea*. Tal componente é um editor simples de textos, semelhante ao consagrado "Bloco de Notas" da empresa Microsoft. O usuário pode tanto digitar diretamente o que deseja como pode colar um texto que esteja na área de transferência de seu computador. Não é possível colar um conteúdo que não seja composto por caracteres, como um imagem, por exemplo. A área de texto é englobada por um painel deslizante (*JScrollPane*) que automaticamente exibe uma barra deslizante em sua borda quando o número de linhas do texto presente ultrapassa a quantidade exibida pela tela.

O botão "Imprimir" (formado por um novo objeto da classe *JButton*) possui a função de invocar os processos de transcrição e impressão do que estiver contido na área de texto. O texto disponibilizado é validado para garantir a certeza daquilo a ser impresso. Caso o texto possua caracteres diferentes dos permitidos ou esteja em branco, um aviso é emitido na tela do programa. O mesmo ocorre quando o microcontrolador está desconectado ou é desligado durante a execução do programa. O usuário também é notificado quando a impressão é finalizada e durante o processo de impressão o botão "Imprimir" e os itens de menu ficam desabilitados.

Enquanto isso, o botão rotulado "Cancelar" possui apenas a função de cancelar uma impressão que ocorre no momento. Esse botão está habilitado somente durante a execução da impressão e a sua atuação implica em uma mensagem para o microcontrolador interromper as suas ações. Quando o cancelamento acontece, um aviso de confirmação é exibido ao usuário e habilita os componentes do programa que estejam indisponíveis. Porém, o cancelamento não é imediato devido à maneira de comunicação entre o computador e o Arduino. Como a interface USB é utilizada, os dados trafegam de forma sequencial (AXELSON, 2009) e são armazenados em um buffer para serem lidos na ordem em que chegam. O cancelamento é representado por uma mensagem e só terá efeito quando for executado pelo microcontrolador.

Apenas os botões e a área de texto apresentados são suficientes para se completar o processo projetado de impressão de textos. No entanto, outras funções foram implementadas de modo a oferecer praticidade e ajuda para o usuário do programa. Dentro do menu "Arquivo" se encontra o item de menu "Novo". Ao se clicar nesse item o usuário é indagado se deseja prosseguir e em caso positivo a área de texto é limpada, ou seja, o texto presente é excluído.

O item de menu “Sair”, também no mesmo menu, oferece ao usuário terminar a execução do programa. Quando clicado exibe uma janela com uma pergunta sobre a certeza desse comando e em caso positivo encerra o programa. O mesmo procedimento ocorre quando se tenta fechar a janela da aplicação pelo botão em seu canto superior direito.

O menu “Ajuda” contém os itens de menu “Ajuda do Impressora Braille” e “Sobre”. O primeiro exibe informações sucintas de como utilizar o programa corretamente. Informa o básico para se inserir um texto e imprimi-lo. Já o segundo apresenta os créditos do corrente programa, como a justificativa para sua montagem.

A tela do programa é montada com um *JFrame* composto por painéis *JPanel*. Todos os itens de menu estão contidos em menus e esses últimos em uma barra de menus. A barra de menus é um componente *JMenuBar*, os menus são componentes *JMenu* e os itens de menu são componentes *JMenuItem* (todos pertencentes ao pacote *Swing*). Enquanto isso, todas as mensagens são construídas com métodos da classe *JOptionPane*. Todos esses componentes estão aprofundados em Deitel (DEITEL, 2006). As figuras 4.5, 4.6 e 4.7 ilustram a tela principal, janelas de informações e mensagens do programa Impressora Braille.

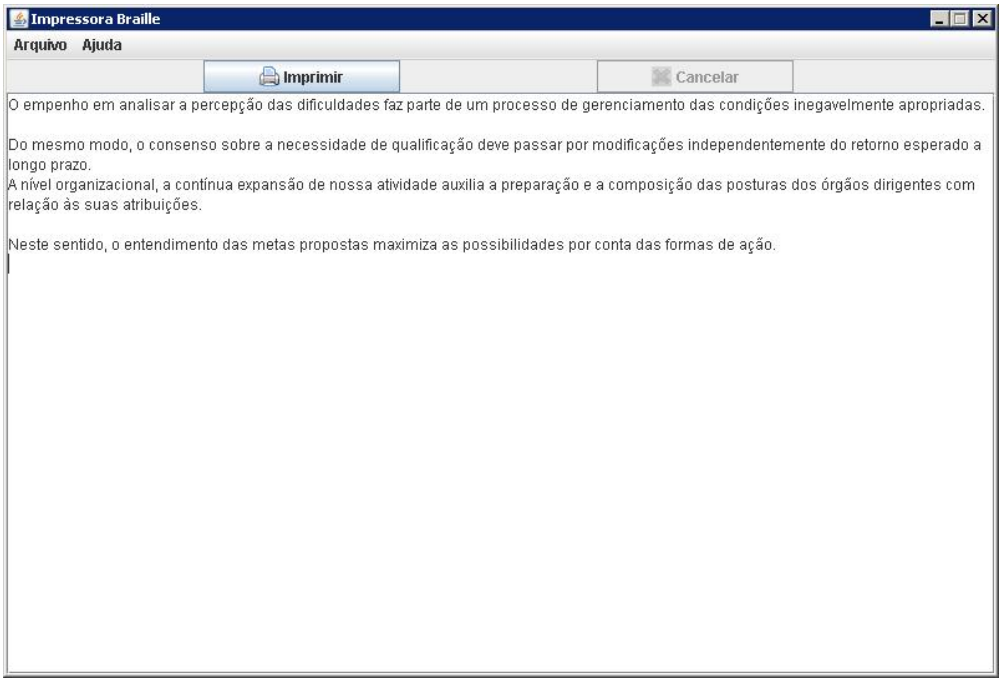

Figura 4.5 – Tela do programa Impressora Braille com um texto inserido. (Fonte: Autor)

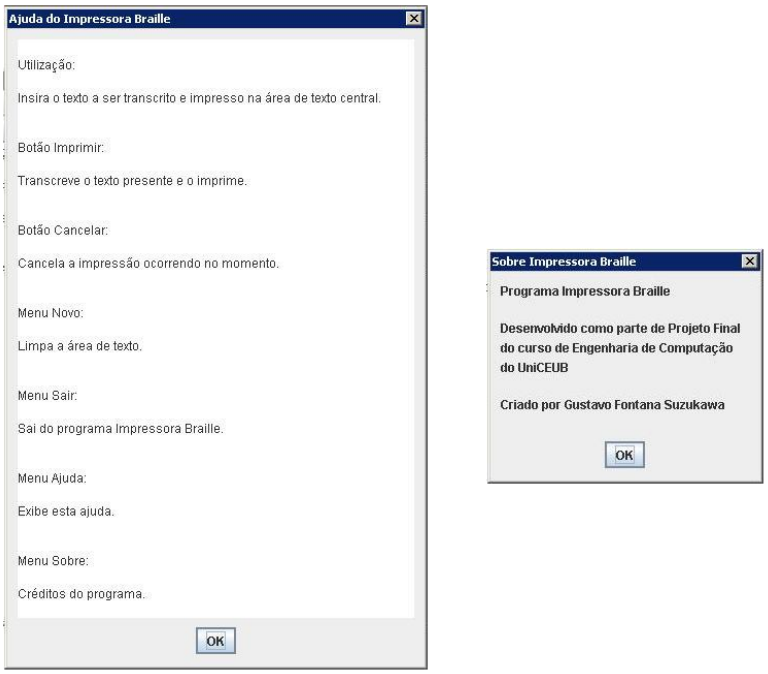

Figura 4.6 – Janelas de informações do programa. (Fonte: Autor)

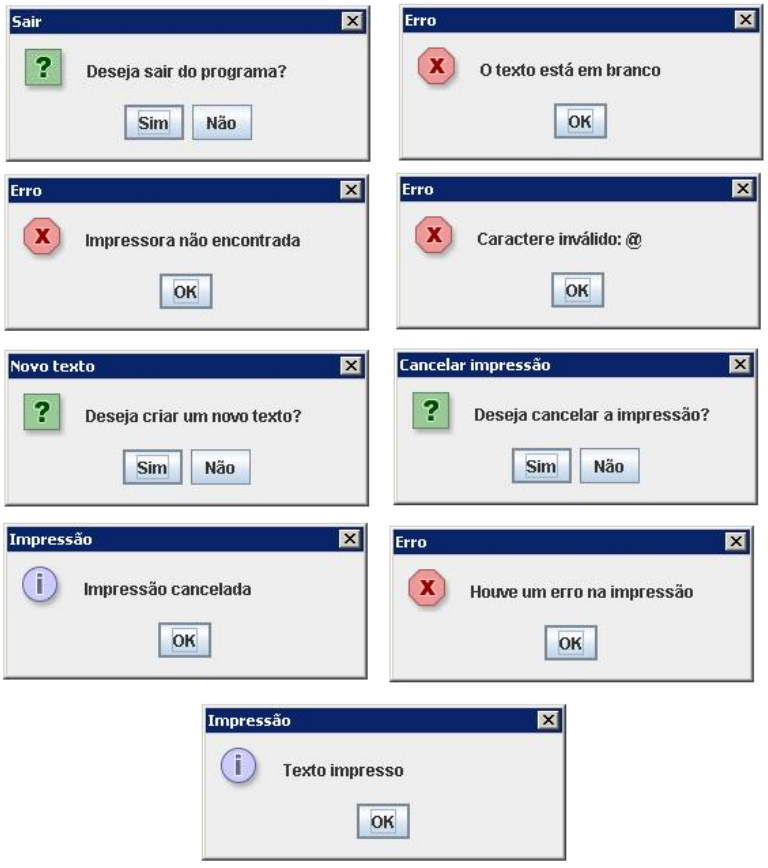

Figura 4.7 – Mensagens exibidas pelo programa. (Fonte: Autor)

4.2.4 – Módulo de transcrição de textos

A transcrição dos textos desejados é uma função desempenhada pela classe Java TraducaoBraille. No entanto, há também uma interação com a classe Impressora por esta ser a responsável pela emissão das informações necessárias ao usuário do programa.

A etapa de transcrição de textos é vital para uma correta impressão visto que seu resultado será representado em uma folha de papel. Toda lógica de transcrição se encontra no computador e não no microcontrolador. Esse último é apenas responsável por executar aquilo que lhe é requisitado.

Para que a classe TraducaoBraille seja capaz de cumprir sua função, ela deve abordar todas as possibilidades do sistema Braille que foram definidas pelo projeto, e isso inclui caracteres variados. Relativamente às outras classes programadas, a classe TraducaoBraille é extensa em número de linhas de código, consequência de sua importância. É importante salientar que em caso de encontrar-se algum caractere que não seja coberto pelo desenvolvimento durante o processo de transcrição, faz-se uso de uma exceção Java criada para tal uso e denominada de ExcecaoCaractereDesconhecido.

4.2.4.1 – Tratamento de caracteres

Um texto é composto por caracteres de diversos tipos diferentes, como alfabéticos, numéricos, de pontuação, de espaçamento, entre outros. Os textos que devem ser transcritos pela aplicação não fogem dessa regra e precisam ter reconhecidos cada um desses caracteres.

Implementar em linhas de código o reconhecimento de caracteres isolados é uma tarefa simples e descomplicada, porém o cenário muda quando se precisa interpretar as palavras formadas por esses caracteres. Algumas regras do sistema Braille existem para cobrir casos particulares na escrita e alteram o texto em questão adicionando mais células ao resultado durante a transcrição do mesmo. Como consequência, a relação de um caractere de texto para cada célula do sistema Braille se extingue (ABREU, 2008). Uma letra maiúscula, por exemplo, no sistema Braille é representada por duas células distintas.

Na transcrição de uma palavra com todas as letras maiúsculas é feita a correspondência de cada letra com a célula adequada e ainda se acrescentam outras duas células sinalizando palavra totalmente maiúscula. Para o caso de números é posta uma célula indicativa de números antes do primeiro número. Se for um número ordinal os valores do sistema Braille mudam para cada algarismo. No entanto é válido lembrar que o programa "Impressora Braille" não realiza nenhuma forma de correção ortográfica ou sintática no texto recebido.

Para transcrever o texto é utilizado um vetor de objetos da classe EstruturaAgulhas. Tal classe possui atributos nomeados de acordo com as posições dos pontos em uma célula Braille (variáveis um, dois, três, quatro, cinco, seis) e se um deles estiver ativado significa que aquele ponto deve ser impresso. O Arduino reconhece tal comando e aciona a respectiva agulha na impressora para realizar a impressão e tornar aquele ponto na folha um área de relevo elevado. Por exemplo toma-se a letra 'a', a qual é representada no sistema Braille pelo ponto número um ativo e os demais inativos em sua célula. O objeto EstruturaBraille desse caractere terá o atributo 'um' com o valor "*true*" (verdadeiro) e os outros "*false*" (falso). Na figura abaixo seguem exemplos de transcrição.

Tabela 4.1 – Exemplos de transcrição

Caractere	Célula Braille	Valores dos atributos					
		Um	Dois	Três	Quatro	Cinco	Seis
a	⠁	true	false	false	false	false	false
z	⠵	true	false	true	false	true	true
-	⠤	false	false	true	false	false	true
&	⠯	true	true	true	true	false	true

Fonte: Autor

O vetor é inteiramente preenchido com vários desses objetos, cada um relativo à uma célula, e todo o texto recebido é analisado dessa maneira para formar a sua representação no sistema Braille.

4.2.5 – Módulo de comunicação

A comunicação entre o computador e o Arduino é realizada através da interface USB e possui a função de interligar o programa “Impressora Braille” e o código sendo executado pelo microcontrolador ATmega2560. Através dela são passados o texto em Braille a ser impresso e os comandos para tornar isso possível.

A classe responsável por tal comunicação chama-se SerialTest e foi adaptada de uma previamente existente (ARDUINO PLAYGROUND, 2010). A classe Impressora cria um novo objeto dessa primeira classe durante a sua execução, mais precisamente após a conclusão da etapa de transcrição de textos. Com o vetor de estruturas preenchido é iniciada uma conexão USB com o microcontrolador, bem como seus respectivos fluxos de entrada e saída de dados.

Se durante o período em que a conexão estiver estabelecida acontecer algum problema de entrada ou saída de dados, é lançada uma exceção *IOException* e a impressão é cancelada. Ao término da impressão os recursos alocados (tempo de processador, memória e interface USB) são liberados pela aplicação.

4.2.6 – Módulo de interpretação de comandos

Todos os comandos possíveis que a impressora deve executar são repassados pelo Arduino, que por sua vez os recebe do computador. Imprimir uma célula, mover a cabeça de impressão ou rolar a folha são tarefas que devem ser interpretadas para serem postas em prática.

Para evitar o risco de o *buffer* USB de entrada do ATmega2560 “estourar” trabalha-se com o limite de 128 *bytes* enviados a cada vez. Cada célula enviada pelo computador possui o tamanho de um *byte*, logo podem ser enviadas 128 células consecutivas sem a necessidade de leitura delas pelo microcontrolador. Como medida de segurança sempre são enviadas 120 células por vez pelo computador, o microcontrolador lê todas elas, limpa o buffer e então requisita mais 120 delas. Isso é feito até o final da impressão desejada.

4.2.6.1 – Formação de comandos

O tipo de variável mais adequado para armazenar e transmitir os comandos necessários do projeto certamente é o *byte*. Ele possui uma dimensão ideal para representar todas as possibilidades de células possíveis e ocupa um espaço reduzido no *buffer* do microcontrolador.

Como um byte pode representar 256 possibilidades distintas, 256 células diferentes podem ser interpretadas pelo microcontrolador. As possibilidades utilizadas no projeto são poucas e englobam caracteres em Braille e comandos para os componentes da impressora. A classe Java EstruturaBraille implementa um célula dessas.

No caso do atributo "outrasFuncoes" da classe EstruturaBraille estar como falso, significa que aquela célula representa um caractere e deve ser impressa. Caso contrário representa um comando a ser avaliado que não seja um caractere. Na tabela 4.2 segue a lista dos comandos disponíveis.

Tabela 4.2 – Mapeamento de uma estrutura

Função (caractere ou tarefa)	EstruturaAgulhas (célula)						
	um	dois	tres	quatro	cinco	seis	outrasFuncoes
Caractere 'c'	true	false	false	true	false	false	false
Caractere 'p'	true	true	true	true	false	false	false
Fim de texto	true	false	false	false	false	false	true
Mover cabeça de impressão para esquerda	false	true	false	false	false	false	true
Mover rolete para mudança de linha	false	false	true	false	false	false	true
Mover rolete para mudança de página	false	false	false	true	false	false	true
Quebra de linha em Java	false	false	false	false	false	true	true

Fonte: Autor

Sempre que a célula deve ser impressa os atributos “um”, “dois”, “três”, “quatro”, “cinco” e “seis” representam os respectivos pontos em Braille que formam um determinado caractere. Por exemplo, uma estrutura com os atributos “outrasFuncoes” como ‘false’ e “um” como ‘true’ impõe que deve ser impressa a letra ‘a’ em Braille na folha. Já “outrasFuncoes” como ‘true’ e “um” como ‘true’ significa o fim do texto a ser impresso.

Com o atributo “outrasFuncoes” como ‘true’ significa um comando a ser interpretado, e eles não são numerosos. “Quebra de linha em Java” representa uma quebra de linha inserida pelo usuário no texto e está implementado somente no código Java, não no microcontrolador. Em uma etapa posterior essas quebras são substituídas pelo comando “Mover rolete para mudança de linha”.

O comando “Fim de texto” é repassado ao microcontrolador quando não há mais caracteres a serem impressos. “Mover cabeça de impressão para esquerda” faz a cabeça de impressão ser movida para a esquerda o tempo necessário para se colocar na posição inicial da linha de impressão. O comando “Mover cabeça de impressão para direita” só está implementado no código do microcontrolador, pois toda impressão de um caractere obrigatoriamente é seguida de um movimento da cabeça de impressão para a direita.

“Mover rolete para mudança de linha” é usado quando o número de caracteres a ser impresso em uma mesma linha atingiu o limite de vinte e cinco caracteres e obriga um movimento do rolete pré-programado equivalente à distância de uma linha de texto no papel. “Mover rolete para mudança de página” é semelhante, mas para uma quantidade de vinte e uma linhas e tempo de movimento do rolete mais prolongado.

O texto a ser impresso está contido em um vetor de EstruturaAgulhas, que obviamente não está definido no código do Arduino. Portanto, cada uma dessas estruturas é convertida em um *byte* onde cada atributo corresponde a um número fixo. Após essa conversão, soma-se os valores da estrutura e o valor (uma variável do tipo *byte*) é enviado ao microcontrolador. Por exemplo, a impressão da letra ‘c’ recebe o valor 18, por ser a soma de 2 com 16 (a letra ‘c’ possui os pontos 1 e 4 ativos na célula Braille). A tabela 4.3 lista as possibilidades de valores dos atributos de uma estrutura.

Tabela 4.3 – Valor dos atributos de uma estrutura

Atributo como ‘true’	um	dois	tres	quatro	cinco	seis	outrasFuncoes
Respectivo valor numérico	2	4	8	16	32	64	1

Fonte: Autor

Dessa maneira cada situação de impressão de caractere ou comandos para impressora é representada por um único valor numérico, a ser interpretado no código presente no microcontrolador ATmega2560.

4.2.6.2 – Obtenção de comandos

Após a transmissão dos bytes para o Arduino, eles são armazenados em um *buffer* e lidos um a um. Primeiro é lido o bit menos significativo, se este estiver setado significa que aquele byte possui um comando a ser executado, caso contrário se trata de um caractere a ser impresso. Existe um tratamento para cada comando a ser executado, ativando o respectivo componente requisitado da impressora (esta fase será melhor explicitada na seção referente ao *hardware* do projeto).

Para imprimir um caractere, uma série de divisões são calculadas. Uma vez que o valor do *byte* contém informações sobre quais agulhas devem ser acionadas para imprimir o caractere desejado, ele deve ser convertido (o processo inverso do que ocorre no computador). O valor do *byte* está armazenado em código decimal e é preciso convertê-lo para binário através de sucessivas divisões pelo número dois (simples conversão de um valor numérico entre bases, nesse caso da decimal para a binária, onde substrai-se o bit menos significativo). Com as divisões efetuadas, tem-se o mesmo valor inicial agora em dimensão de *bits*, sendo cada um deles relativo a um ponto da célula Braille, a uma agulha que deve ser acionada.

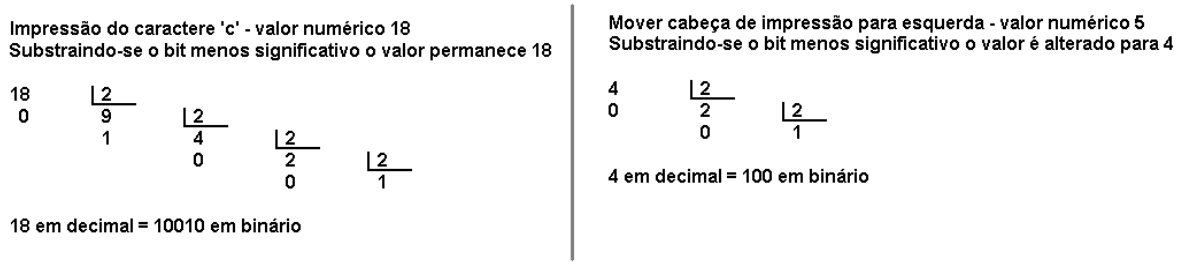

Figura 4.8 – Exemplos de conversão. (Fonte: Autor)

Logo que todos os *bytes* no *buffer* são recebidos, interpretados e executados, mais 120 *bytes* são enviados para o microcontrolador, em uma tarefa que se esgota somente quando todo o texto transcrito é impresso. Para indicar fim de texto o último *byte* recebido é um comando do tipo "Fim de texto". Assim, a aplicação cumpre seu papel e está pronta para realizar uma nova impressão.

4.3 – Hardware

O *hardware* desenvolvido para o projeto inclui um microcontrolador ATmega2560, a placa onde esse se encontra (Arduino), quatro CIs (circuitos integrados) L298 e uma impressora matricial. Tal impressora têm como componentes envolvidos um rolete para movimentar a folha a ser impressa, um carro de impressão onde se localiza a cabeça de impressão e a própria cabeça.

Há um motor de passo para controlar o rolete e outro para o carro de impressão. As agulhas da cabeça de impressão são constituídas de solenóides que são ativados quando uma tensão conhecida é aplicada em seus terminais.

4.3.1 – Interface microcontrolador/impressora

Toda a lógica de impressão implementada no projeto é realizada por um circuito desenvolvido pelo autor, ou seja, quase nada é aproveitado do circuito existente na impressora Epson, apenas pinos de alimentação de 5V e 24V (Volts) e terra. Portanto, se faz necessário

controlar dois motores de passo e seis agulhas existentes através do microcontrolador e demais CIs auxiliares.

4.3.1.1 – Circuito integrado L298

Este circuito integrado é voltado para o controle de motores de passo devido à sua alta confiabilidade. Simplesmente é alimentado com tensões de 5V e 24V (mesma tensão do motor) e gera saídas em nível alto conforme as entradas recebidas do microcontrolador. Suas quatro saídas são ligadas às quatro entradas do motor de passo e em uma sequência programada com o objetivo de ocasionar a rotação de tal motor. As suas entradas são configuradas pelo ATmega2560, o qual fornece 5V em determinada entrada para ativar a respectiva saída em 24V. A figura 4.9 mostra um exemplar do CI L298.

Figura 4.9 – Circuito integrado L298. (Fonte: http://store.robolution.co.in/index.php?main_page=product_info&cPath=10_13&products_id=6)

4.3.1.2 – Comunicação com os componentes

As portas de quarenta e seis a quarenta e nove da placa Arduino são configuradas para servirem de entrada de um CI L298, o qual possui suas quatro saídas conectadas ao motor de passo do rolete da impressora. Da mesma maneira as portas de cinquenta a cinqüenta e três servem de entrada para outro L298, com suas saídas para o motor de passo do carro de

impressão. Um terceiro CI igual aos anteriores é conectado ao Arduino pelas portas de vinte e dois a vinte e quatro e é responsável pelo acionamento das agulhas um, dois e três. Por fim, um último L298 se comunica pelas portas de vinte e cinco a vinte e sete e controla as agulhas quatro, cinco e seis.

4.3.2 – Impressora matricial

A impressora possui uma placa de circuito impresso para impressão de folhas a partir de um computador, porém ela não satisfaz as necessidades do projeto e por tal razão é necessário confeccionar uma placa específica para os componentes existentes. Tais componentes são o rolete, o carro e a cabeça de impressão.

4.3.2.1 – Rolete

Existe somente um rolete na impressora, com a função de tracionar a folha onde deve ser impresso o texto transcrito para o sistema Braille. É movimentado através de um motor de passo e está conectado a este por meio de uma engrenagem. Conforme o motor é acionado por um CI L298 o rolete entra em movimento contínuo ou intermitente, de acordo com o necessário.

É composto por uma haste metálica encoberta por uma capa plástica resistente e não flexível. Essa última está envolvida por uma esponja com a função de amortecer o impacto das agulhas sobre a folha. A haste está conectada à engrenagem anteriormente mencionada e é rotacionada no mesmo sentido do motor de passo. O rolete da impressora está ilustrado na figura 4.10.

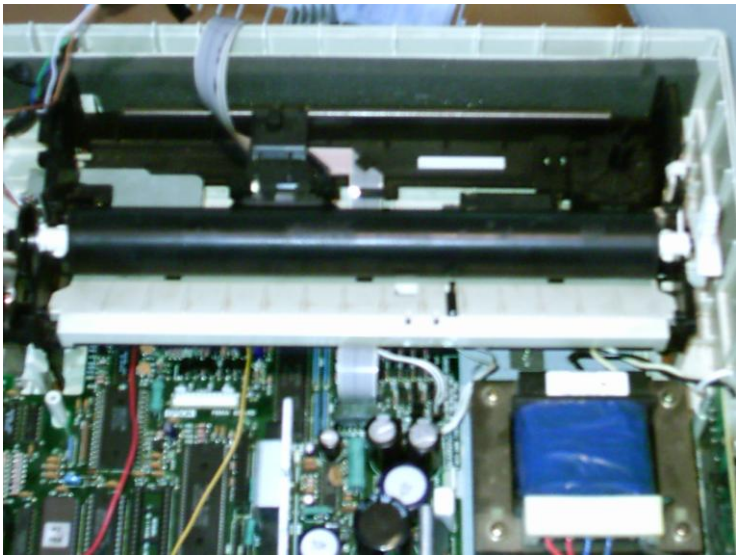

Figura 4.10 – Rolete da impressora Epson. (Fonte: Autor)

4.3.2.2 – Carro de impressão

O carro de impressão é o componente responsável por guiar a posição horizontal das células braille no papel. É deslocado sobre duas hastes paralelas e controlado por movimentos oriundos de um motor de passo específico. Tal motor possui a capacidade de rotacionar seu eixo nos sentidos horário e anti-horário, ocasionando o movimento do carro de impressão para a direita e para a esquerda sobre as hastes.

Seu respectivo motor é bastante preciso, assim como aquele conectado ao rolete. É possível obter deslocamentos mínimos do carro de impressão com acionamentos curtos do seu motor de passo, lembrando que um CI L298 controla esse último através de suas saídas. O carro de impressão é mostrado na figura 4.11.

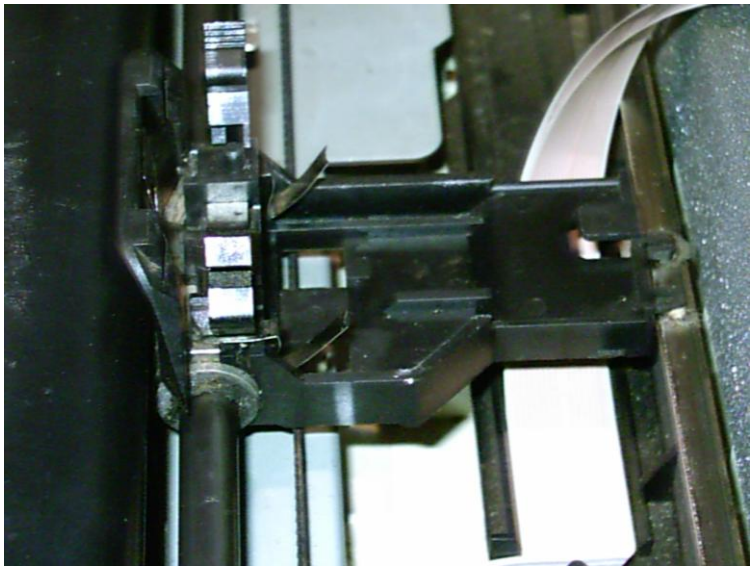

Figura 4.11 – Carro de impressão existente na Epson. (Fonte: Autor)

4.3.2.3 – Cabeça de impressão

A cabeça de impressão foi totalmente reformulada em relação àquela existente originalmente. Foram incluídos solenóides, agulhas comercializadas em farmácias (para simular o instrumento de escrita braille denominado punção) e construídos guias para as agulhas. Na extremidade de cada solenóide foi acoplada uma agulha e todos os solenóides estão dispostos paralelamente em duas colunas, representando a célula Braille. Esta cabeça permanece sobre o carro de impressão a todo momento.

Para manter a correta direção das agulhas foram construídos guias vazados por onde elas passam. Com o acionamento das agulhas desejadas a folha corrente é impactada e deformada. As figuras 4.12 e 4.13 apresentam a cabeça de impressão vista por dois ângulos distintos.

Figura 4.12 – Cabeça de impressão desenvolvida. (Fonte: Autor)

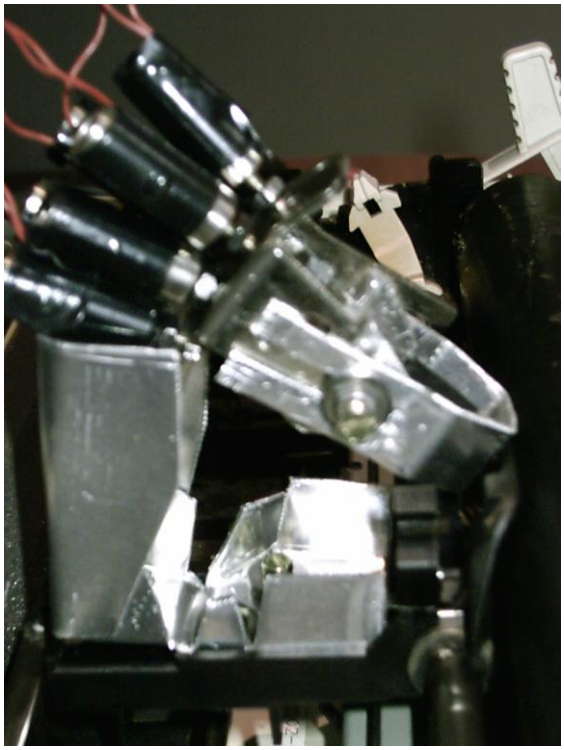

Figura 4.13 – Cabeça de impressão por outro ângulo. (Fonte: Autor)

4.3.2.4 – Atuação dos componentes

Os componentes da impressora dependem de motores de passo e solenóides para atuarem corretamente. Devem ser acionados de maneira confiável e para tal controlados pelo microcontrolador e demais circuitos integrados disponíveis.

O rolete e o carro de impressão são movimentados por motores de passo unipolares, sempre com passos completos. Tais motores são ativados por um curto período de tempo e logo em seguida desativados para evitar superaquecimento de sua estrutura. Pelo fato de possuírem bobinas em seu interior por onde passa uma corrente considerável, não devem estar em funcionamento contínuo por poderem ser inutilizados. Cada motor está conectado a um circuito integrado L298, pelo qual recebe comandos de deslocamento.

A cabeça de impressão engloba os solenóides que acionam as agulhas destinadas a impactarem a folha onde deve ser impresso o texto. Outros dois CIs L298 são conectados à cabeça, um para cada grupo de três solenóides. A configuração desses CIs é semelhante à dos anteriores, porém as suas tensões de alimentação são de 5V e 12V e esta última tensão também é provida para cada solenóide.

Uma consideração importante a ser feita é o espelhamento do texto a ser impresso. Em uma impressora matricial comum a cabeça de impressão impacta uma folha através uma fita contendo tinta pressionada por diversas agulhas, ou seja, o texto está voltado para a cabeça de impressão. Após a adaptação da impressora para o projeto as agulhas começaram a deformar a folha ao invés de marcá-la com tinta, assim o texto deixou de estar voltado para a cabeça de impressão. Tal mudança alterou a posição de descanso dessa cabeça para o lado contrário das hastes sobre as quais ela desliza e a folha na qual é impresso o texto deve ser posicionada no outro extremo do rolete se comparado à situação anterior.

Dessa maneira, todos os componentes que constituem a placa do circuito estão definidos e é possível representar suas conexões através do programa de computador especializado para tal fim denominado *Proteus*, como mostrado nas figuras 4.14 e 4.15. Após essa etapa a placa em si pode ser confeccionada e interligar definidamente todos os módulos da solução desenvolvida. As figuras 4.16 e 4.17 ilustram a placa de circuito impresso.

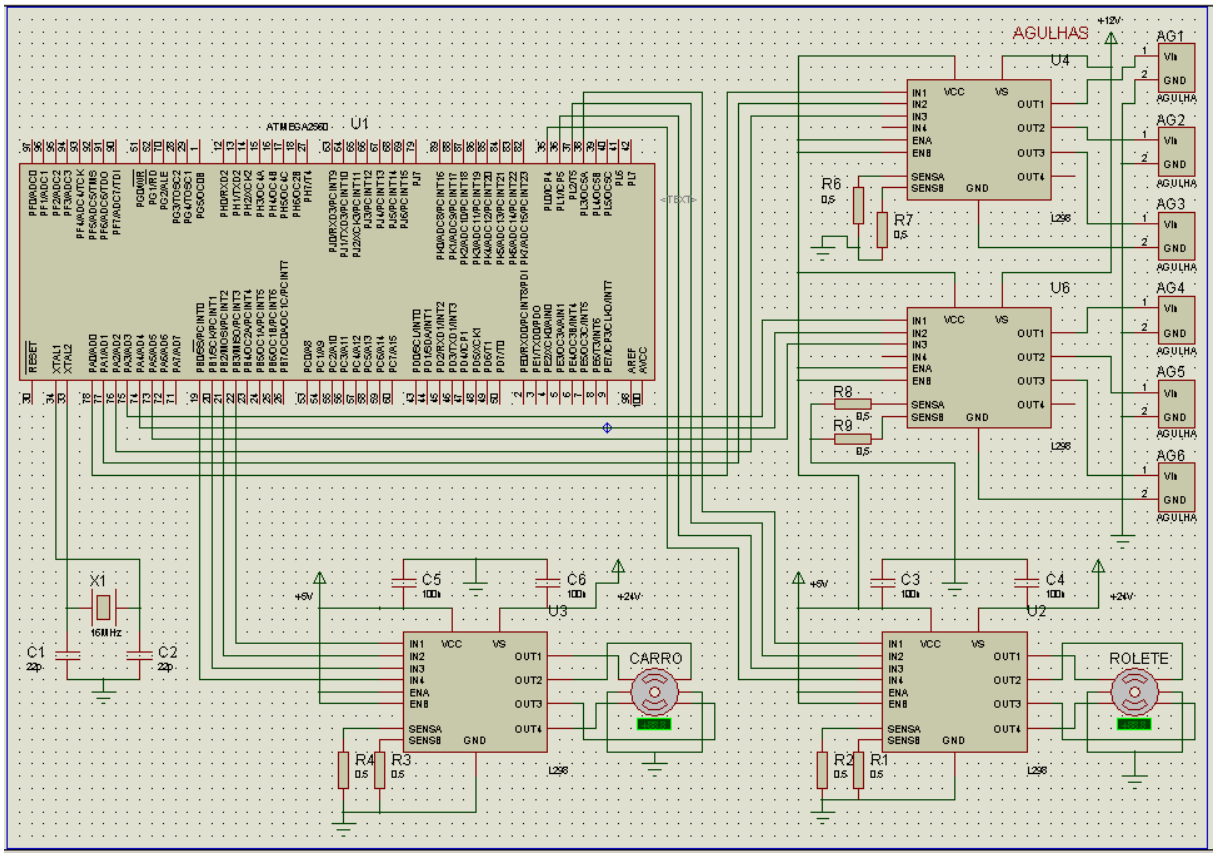

Figura 4.14 – Representação do circuito projetado no software Proteus. (Fonte: Autor)

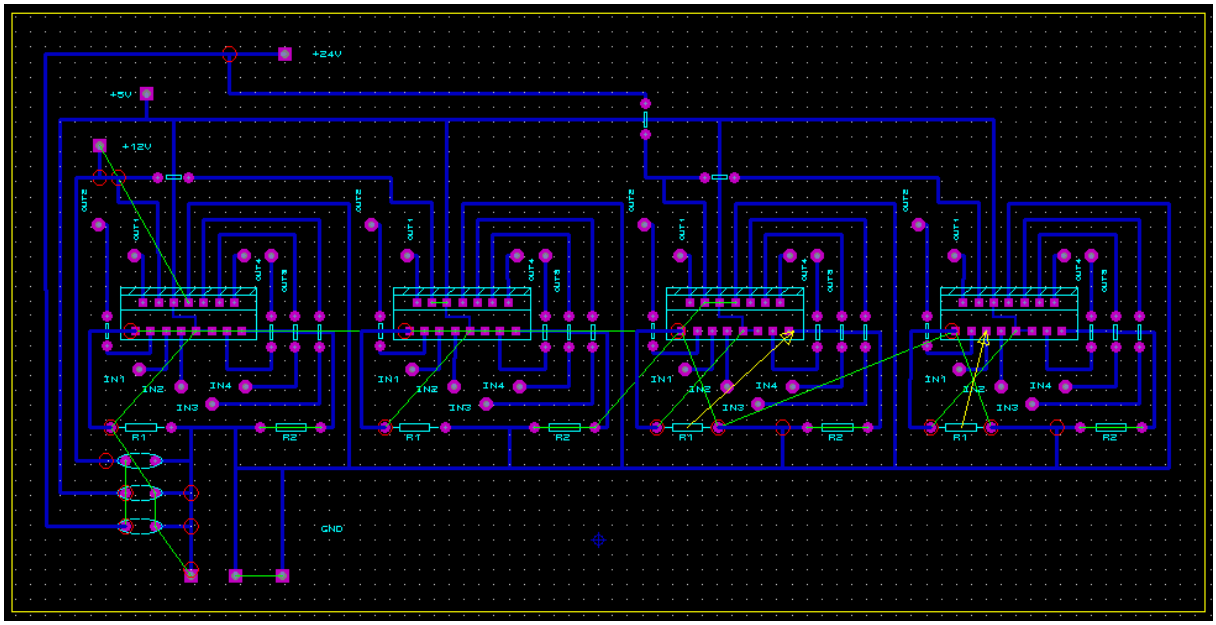

Figura 4.15 – Projeto da placa de circuito impresso no Proteus. (Fonte: Autor)

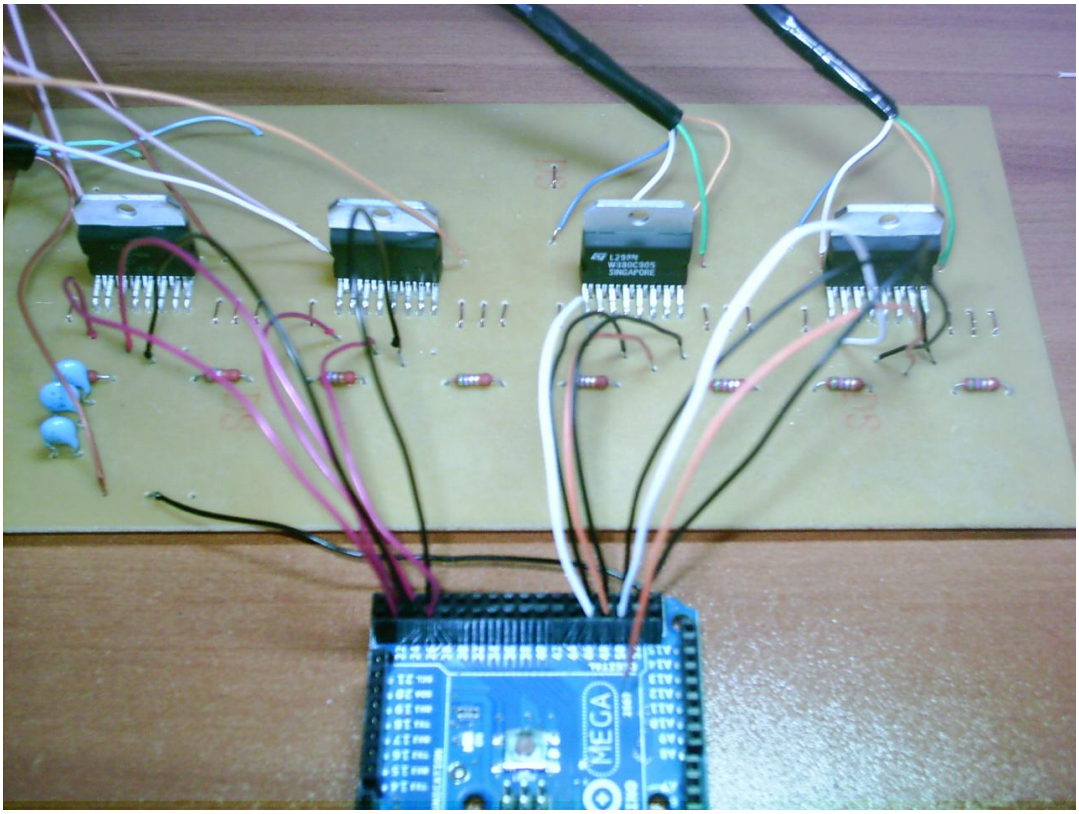

Figura 4.16 – Placa de circuito impresso em estágio final. (Fonte: Autor)

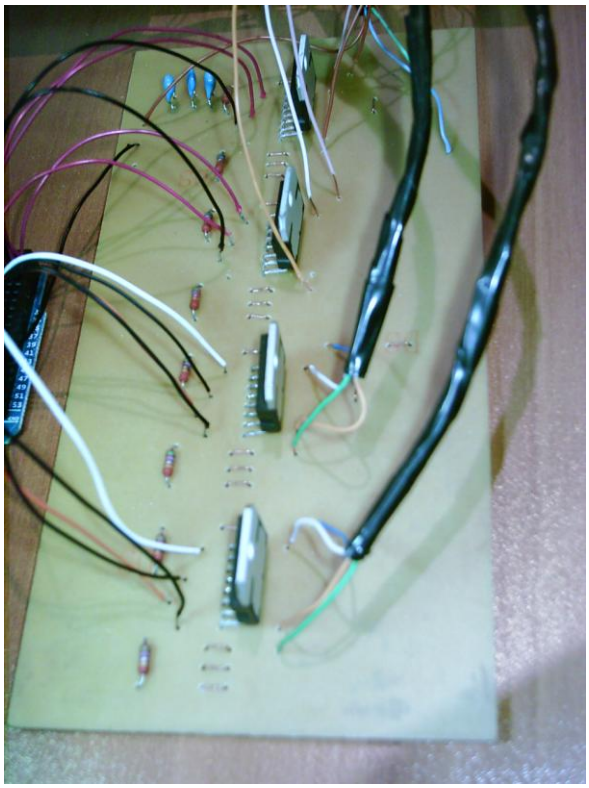

Figura 4.17 – Vista lateral da placa finalizada. (Fonte: Autor)

CAPÍTULO 5 – APLICAÇÃO DO MODELO PROPOSTO

5.1 – Área de Aplicação

A área de aplicação da solução desenvolvida é representada pela verificação dos resultados por parte do autor. É considerado válido o resultado do projeto se a impressora for capaz de controlar seus motores corretamente, assim como as agulhas presentes na cabeça de impressão com a finalidade de imprimir o que lhe for demandado.

5.2 – Descrição da Aplicação do Modelo

Inicialmente o texto “Energias renováveis contra o aquecimento global”, o qual se encontra no anexo B, é inserido no programa de computador Impressora Braille e ordena-se a sua impressão. Ela é realizada e seu resultado é avaliado pelo autor para verificar se os comandos executados estavam de acordo com o programado.

Ao se ordenar a impressão o texto é transcrito para o sistema Braille e enviado ao microcontrolador. Este interpreta os comandos necessários e efetivamente imprime o texto desejado, mas com ressalvas explicitadas adiante.

5.3 – Resultados

5.3.1 – Resultados esperados

Em relação ao *software* espera-se que a aplicação transcreva os textos de maneira correta, ou seja, de acordo com o as regras do sistema Braille. Também deve informar em caso de caracteres desconhecidos presentes no texto, assim como detectar falhas de comunicação entre o computador e o microcontrolador. Quando desejado o cancelamento de uma impressão

corrente, o programa precisa fazê-lo. Por fim, é necessária a correta transmissão do texto transcrito para o microcontrolador.

Quanto ao *hardware* as expectativas são de que o Arduino precisa receber satisfatoriamente o resultado da transcrição do texto ordenado, interpretar os comandos relativos à impressão e efetivamente coordenar o processo de marcar na folha de papel os caracteres em Braille. Já a impressora deve ser integrada por componentes confiáveis e que atendam as demandas de impressão.

5.3.2 – Resultados obtidos

A aplicação foi capaz de transcrever o texto proposto para o sistema Braille e transmiti-lo ao microcontrolador para posterior impressão. O microcontrolador ATmega2560 interpretou corretamente os comandos recebidos e ativou os circuitos integrados necessários. Tais CIs forneceram tensões nos momentos previstos e acionaram os motores e agulhas de maneira precisa, mas a impressão do texto não atendeu todas as especificações planejadas anteriormente.

A cabeça de impressão é formada por seis solenóides provenientes de um equipamento denominado teleimpressor (modelo Olivetti) que foram acopladas à sua estrutura. Porém, tais solenóides não são os ideais para a solução desenvolvida porque seu deslocamento axial é minúsculo e eles não aplicam uma força suficiente para deformar o papel com gramatura de $120 g/m^2$. Além disso, as agulhas previamente existentes nos solenóides se dobram facilmente quando entram em contato com outro material. Antes disso punções foram utilizados nas extremidades dessas agulhas, mas possuíam um comprimento reduzido e eram de difícil manuseio, fatos que causaram seus descarte.

Logo, para demonstrar que o processo de impressão está correto, foi impresso o texto do anexo B com agulhas de farmácia para simular os punções em uma folha com gramatura de $75 g/m^2$. Torna-se complicado para um deficiente visual realizar a leitura do conteúdo da folha porque os relevos produzidos foram pequenos, mas substituir os solenóides utilizados por outros mais adequados garantiria a leitura por parte dos deficientes visuais. Para tornar mais fácil a visualização do resultado da impressão, os pontos onde a folha foi marcada em relevo

foram destacados com o uso de grafite. Os pontos impressos são identificados como os pequenos pontos negros na imagem da figura 5.1.

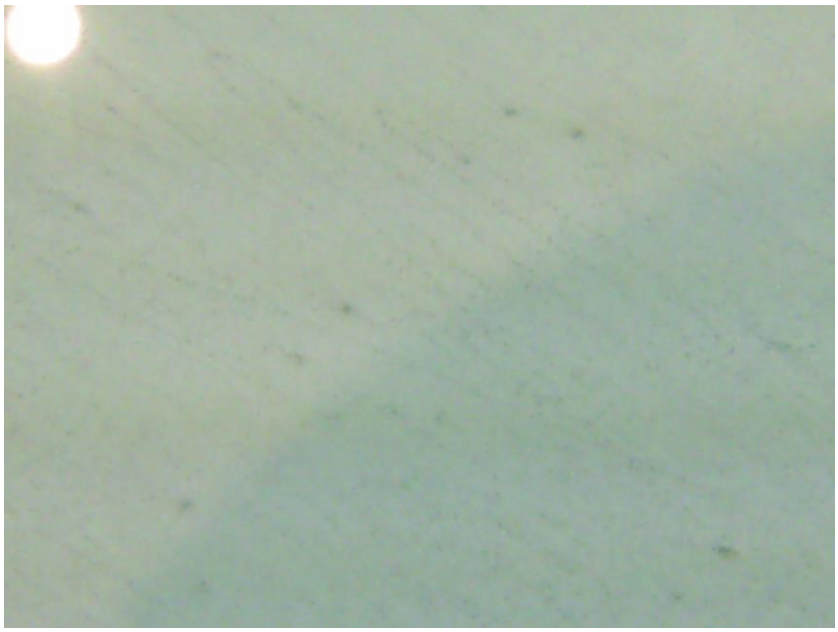

Figura 5.1 – Folha impressa com texto em Braille pela solução. (Fonte: Autor)

5.3.3 – Comparação entre os resultados

O *software* atendeu às expectativas do autor ao transcrever corretamente os textos que lhe foram ordenados, além de transmiti-los sem erros para o microcontrolador.

O *hardware* também foi quase totalmente completado dentro das especificações, não fosse pelos solenóides utilizados. Para agrupá-los ainda foi desenvolvida uma estrutura em alumínio, mantendo-os coesos e direcionados à folha a ser impressa. O microcontrolador, o circuito desenvolvido e os componentes da impressora atuaram corretamente, assim como era esperado.

Em outras palavras, o projeto é capaz de produzir impressões em Braille, porém não atende as especificações relativas às medidas da cela braille, como a profundidade e o diâmetro dos relevos. Isso se deve aos solenóides utilizados, mas que fazem parte de áreas pertencentes à Engenharia Mecânica e não à Engenharia da Computação. Dimensionar os solenóides corretos a serem integrados e montar uma estrutura física adequada aos mesmos demandam do

autor conhecimentos não adquiridos durante o período de curso. Os demais componentes do projeto são de domínio prévio do autor e funcionam como esperado.

5.4 – Custos

Um desejo por parte do autor antes e durante o desenvolvimento do projeto era diminuir custos para se obter uma impressora Braille. De uma maneira geral isso foi possível analisando-se separadamente cada componente utilizado.

O *software* transcritor não possui custo nenhum, é gratuito. O *Java Development Kit* e o driver do Arduino também não custam nada. A placa Arduino Mega 2560 possui um valor de R$ 315,00 e o cabo USB A/B custou R$ 3,90. Também foram utilizados quatro CIs L298, cada um custando R$ 5,00. Os quatro CIs totalizaram uma despesa de R$ 20,00.

A impressora matricial Epson LQ-500 foi doada para o autor porque a sua finalidade de uso era acadêmica. Devido à sua longínqua data de fabricação ela não se encontra mais disponível no mercado. No entanto, a sua sucessora denominada Epson LQ-590 custa US$ 399,00 (diretamente do fabricante), algo em torno de R$ 665,00.

As agulhas obtidas em farmácias custaram um total de R$ 10,00. Logo, o projeto custou um total de R$ 348,90 e para reproduzi-lo seriam gastos R$ 1013,90 considerando-se substituir a impressora do projeto por sua sucessora.

Dentre as impressoras Braille citadas no capítulo de apresentação do problema a que possui um custo menor é a *Basic D* com um valor aproximado de R$ 5354,00, além da taxa cobrada para frete e importação porque ela é vendida pelo fabricante (lembrando que todos os fabricantes são estrangeiros). Tal impressora possui mais recursos se comparada à impressora utilizada no projeto. Considerando-se um programa transcritor gratuito para ser usado junto à ela, pode-se traçar uma comparação dos custos envolvidos, conforme mostra a tabela 5.1.

Apesar das limitações da solução desenvolvida, a diferença entre o seu custo e o custo da alternativa com preço menos elevado é considerável. Reproduzir o projeto custa R$ 4340,10 menos do que adquirir uma impressora disponível no mercado. O custo para reprodução equivale a quase 19% do custo da alternativa comparada.

Tabela 5.1 – Custo total do projeto e comparações

	Software transcritor	Arduino e cabo USB	Circuitos integrados	Impressora matricial	Agulhas	Custo total
Projeto desenvolvido	R$ 0,00	R$ 318,90	R$ 20,00	R$ 0,00	R$ 10,00	R$ 348,90
Reprodução do projeto	R$ 0,00	R$ 318,90	R$ 20,00	R$ 665,00	R$ 10,00	R$ 1013,90
Alternativa do mercado	R$ 0,00	R$ 0,00	R$ 0,00	R$ 5354,00	R$ 0,00	R$ 5354,00

Fonte: Autor

5.5 – Avaliação Global

De uma maneira geral o projeto cumpriu seu objetivo de produzir uma impressora para textos em Braille na língua portuguesa. A solução é capaz de receber e transcrever textos, enviá-los à impressora e imprimir seu resultado. A única ressalva se deve às diferenças entre os espaçamentos e profundidades da cela braille. No entanto, tal ajuste demanda ações que extrapolam os conhecimentos e competências do autor perante o curso de Engenharia de Computação. Substituir os solenóides mencionados por modelos mais adequados e ajustar a estrutura metálica produzida para abrigá-los são os únicos ajustes necessários para que os deficientes visuais consigam ler os textos impressos.

Certamente o *software* transcritor caracteriza-se como um ponto forte do projeto, visto sua aplicação e personalização para a língua portuguesa. O uso de motores de passo também alavanca o sucesso do projeto devido à sua precisão de movimentos. Já a cabeça de impressão foi considerada a parte mais complicada de ser desenvolvida pelo autor, consequência de sua proximidade com a área de atuação da Engenharia Mecânica.

Durante o desenvolvimento do projeto o autor adquiriu vários conhecimentos interessantes e necessários para o sucesso do mesmo, entre eles o funcionamento de motores de passo e técnicas de programação voltadas para a orientação a objeto. Porém o maior ganho

do autor não foi um conhecimento adquirido, mas sim a compreensão das dificuldades de leitura e escrita por parte dos deficientes visuais. Essa percepção serviu de inspiração para a conclusão do projeto.

CAPÍTULO 6 – CONCLUSÃO

6.1 – Conclusões

O desenvolvimento do projeto proposto foi extremamente válido, agregou conhecimentos e pontos de vista diferenciados. Para executar o que havia sido planejado precisou-se de dedicação por longos e frequentes períodos de tempo, seriedade em relação ao tema escolhido e compreensão dos problemas encontrados pelos deficientes visuais na leitura e escrita de textos em Braille.

O objetivo proposto de desenvolver um protótipo de impressora que imprima textos em Braille na língua portuguesa, a partir de uma impressora matricial existente foi cumprido parcialmente, quase totalmente. Os solenóides da cabeça de impressão inviabilizaram a impressão nas especificações.

A aplicação transcritora foi desenvolvida conforme planejado, assim como o circuito que serve de interface entre o microcontrolador utilizado e os mecanismos da impressora adaptada. Textos diversos também foram impressos para validar a solução dentro das suas limitações.

Os resultados obtidos demonstraram que a aplicação e a impressora funcionam, mas essa última é dependente de um componente específico para ser uma solução plena. Os textos impressos foram transcritos para o sistema Braille conforme a Grafia Braille para a Língua Portuguesa, enviados ao microcontrolador e representados em folhas de papel. Apenas as especificações de espaçamento envolvendo as células não foram contempladas.

A relação custo/benefício é favorável à solução desenvolvida. Conforme explicitado em seção anterior a diferença monetária entre a aquisição de uma impressora Braille existente no mercado e a reprodução da solução é de no mínimo R$ 4340,10, visto que se escolheu a alternativa menos custosa para a comparação executada.

6.2 – Sugestões para Trabalhos Futuros

Muito ainda pode ser feito na área de tecnologias assistivas, principalmente quando voltadas às necessidades dos deficientes visuais. Em relação ao projeto desenvolvido um aprimoramento extremamente válido seria dimensionar e acoplar solenóides específicos à cabeça de impressão. Uma vez que a impressora já realiza o planejado corretamente, adequar as agulhas às especificações elevaria a qualidade da solução.

Tornar a impressora e a aplicação independentes do microcontrolador utilizado também melhora a solução, pois abrange um número maior de componentes de *hardware* e pode reduzir custos de produção. Por exemplo, utilizar um microcontrolador da família *PIC* ao invés de um da família ATmega estende a gama de desenvolvedores disponíveis.

Uma terceira maneira de agregar valor ao projeto é substituir o carro de impressão por uma cabeça de impressão que cubra toda a largura da folha a ser impressa. Dessa maneira menos um motor deveria ser controlado e a tecnologia utilizada em um solenóide poderia ser replicada em uma série de outros paralelamente dispostos.

REFERÊNCIAS

ABNT. Norma Brasileira 9050: Acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. 2ª edição. Rio de Janeiro: ABNT, 2004. 97 p.

ABREU, Elza; SANTOS, Fernanda; FELLIPE, Maria; OLIVEIRA, Regina. Braille!? O que é isso?. São Paulo: Fundação Dorina Nowill para Cegos, 2008. 52 p.

ARDUINO PLAYGROUND. Arduino playground - Java. 2010. Disponível em: <http://www.arduino.cc/playground/Interfacing/Java> Acesso em: 07 out. 2010.

ATMEL. ATmega2560 Preliminary Summary. San Jose: 2010. 38 p. Disponível em: <http://www.atmel.com/dyn/resources/prod_documents/2549S.pdf> Acesso em: 12 nov. 2010.

AXELSON, Jan. USB Complete: The Developer's Guide. Madison: Lakeview Research, 2009. 506 p.

BANZI, Massimo. Getting Started with Arduino. 1ª edição. Sebastopol: O'Reilly Media, 2009. 118 p.

BRASIL. Ministério da Educação. Grafia Braille para a Língua Portuguesa. Secretaria de Educação Especial. Brasília: SEESP, 2006. 106 p.

BRITES, Felipe; SANTOS, Vinicius. Motor de Passo. Niterói: 2008. 15 p. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf> Acesso em: 02 nov. 2010.

DEITEL, Paul; DEITEL, Harvey. Java: How to Program. 7ª edição. New Jersey: Prentice Hall, 2006. 1500 p.

EPSON. LQ-500 Impact Printer Product Information Guide. Nagano: 1988. 5 p. Disponível em: <http://files.support.epson.com/pdf/l1000_/l1000_pg.pdf> Acesso em: 07 set. 2010.

FORBELLONE, André; EBERSPÄCHER, Henri. Lógica de Programação: A construção de algoritmos e estruturas de dados. 3ª edição. São Paulo: Prentice Hall, 2005. 218 p.

GRANDI, Antônio; NORONHA, Paulo. Informática e deficiência visual: Uma relação possível?. São Paulo: Fundação Dorina Nowill para Cegos, 2010. 54 p.

GREENPEACE. Energias renováveis contra o aquecimento global. 2010. Disponível em: <http://www.greenpeace.org/brasil/pt/O-que-fazemos/Clima-e-Energia/> Acesso em: 05 nov. 2010.

MACHADO, Rosane; MERINO, Eugenio. Descomplicando a Escrita Braille: Considerações a Respeito da Deficiência Visual. Curitiba: Juruá Editora, 2009. 94 p.

MONTEIRO, Mário. Introdução à Organização de Computadores. 4ª edição. Rio de Janeiro: LTC Editora, 2002. 498 p.

NICOLOSI, Denys. Microcontrolador 8051 Detalhado. 8ª edição. São Paulo: Editora Érica, 2007. 238 p.

REAS, Casey; FRY, Ben. Getting Started with Processing. 1ª edição. Sebastopol: O'Reilly Media, 2010. 208 p.

APÊNDICE A – Classes Java do projeto

1 – Classe Impressora

```
/*
 *      Autor: Gustavo Fontana Suzukawa
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
 *      A classe Impressora apresenta ao usuário uma interface para inserir
 *      o texto que deve ser transcrito para Braille e posteriormente impresso.
 *      Coordena a função de transcrição de textos, bem como a comunicação
 *      com a placa Arduino.
 */

public class Impressora extends JFrame implements ActionListener, WindowListener{
    //Area de texto para receber o texto a ser transcrito e impresso
    private JTextArea areaTexto;
    //Area de texto contendo a ajuda do programa
    private JTextArea areaTextoAjuda;
    //Painel deslizante que contém a área de texto principal
    private JScrollPane painelDeslizante;
    //Barra de menus do programa
```

```
private JMenuBar menu;
//Menu 'Arquivo'
private JMenu menuArquivo;
//Menu 'Ajuda'
private JMenu menuAjuda;
//Item de menu 'Novo'
private JMenuItem menuItemNovo;
//Item de menu 'Sair'
private JMenuItem menuItemSair;
//Item de menu 'Ajuda'
private JMenuItem menuItemAjuda;
//Item de menu 'Sobre'
private JMenuItem menuItemSobre;
//Botão 'Imprimir'
private JButton botaoImprimir;
//Botão 'Cancelar'
private JButton botaoCancelar;
//Painel principal do programa
private JPanel painelPrincipal;
//Painel superior contendo o menu
private JPanel painelSuperior;
//Painel contendo os botões
private JPanel painelBotoes;
//Classe para se comunicar com a placa Arduino
private SerialTest conexaoSerial;
//Classe para transcrever o texto para Braille
```

```
private TraducaoBraille traducao;
//String com instruções mostradas no menu 'Ajuda'
private String instrucoes;
//Opções disponíveis das mensagens exibidas
private Object[] opcoesMensagem = {"Sim", "N\u00e3o"};

/*
 *    Construtor da classe que instancia e ajusta os componentes
 *    do programa para transcrever e imprimir textos
 */
public Impressora(){
    //Título da janela
    super("Impressora Braille");
    //Tamanho da janela
    setSize(800, 540);
    //Localização da janela
    setLocation(200, 100);
    //Por padrão, não fazer nada ao fechar-se a janela
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    //Inicializa a área de texto principal
    areaTexto = new JTextArea();
    //Inicializa a área de texto da ajuda
    areaTextoAjuda = new JTextArea(30, 35);
    //Inicializa a barra de menus
    menu = new JMenuBar();
    //Inicializa o menu 'Arquivo'
```

```
menuArquivo = new JMenu("Arquivo");
//Inicializa o menu 'Ajuda'
menuAjuda = new JMenu("Ajuda");
//Inicializa o item de menu 'Novo'
menuItemNovo = new JMenuItem("Novo");
//Inicializa o item de menu 'Sair'
menuItemSair = new JMenuItem("Sair");
//Inicializa o item de menu 'Ajuda'
menuItemAjuda = new JMenuItem("Ajuda do Impressora Braille");
//Inicializa o item de menu 'Sobre'
menuItemSobre = new JMenuItem("Sobre");
//Inicializa o botão 'Imprimir'
botaoImprimir = new JButton("Imprimir", new ImageIcon("imprimir.gif"));
//Inicializa o botão 'Cancelar'
botaoCancelar = new JButton("Cancelar", new ImageIcon("cancelar.gif"));
//Desabilita o botão 'Cancelar'
botaoCancelar.setEnabled(false);
//Inicializa o painel principal
painelPrincipal = new JPanel(new BorderLayout());
//Inicializa o painel superior
painelSuperior = new JPanel(new BorderLayout());
//Inicializa o painel dos botões
painelBotoes = new JPanel(new GridLayout(1, 5));

//Guarda em uma variável o texto que será mostrado na janela de 'Ajuda' do programa
```

instrucoes = "\nUtiliza\u00e7\u00e3o:\n\nInsira o texto a ser transcrito e impresso " + "na \u00e1rea de texto central.\n\n\nBot\u00e3o Imprimir:\n\nTranscreve o texto " + "presente e o imprime.\n\n\nBot\u00e3o Cancelar:\n\nCancela a impress\u00e3o " + "ocorrendo no momento.\n\n\nMenu Novo:\n\nLimpa a \u00e1rea de texto.\n\n\n" + "Menu Sair:\n\nSai do programa Impressora Braille.\n\n\nMenu Ajuda:" + "\n\nExibe esta ajuda.\n\n\nMenu Sobre:\n\nCr\u00e9ditos do programa.\n";

```
//Configura um evento ao se clicar no botão 'Imprimir'
botaoImprimir.addActionListener(this);
//Configura um evento ao se clicar no botão 'Cancelar'
botaoCancelar.addActionListener(this);
//Configura um evento ao se clicar no item de menu 'Novo'
menuItemNovo.addActionListener(this);
//Configura um evento ao se clicar no item de menu 'Sair'
menuItemSair.addActionListener(this);
//Configura um evento ao se clicar no item de menu 'Ajuda'
menuItemAjuda.addActionListener(this);
//Configura um evento ao se clicar no item de menu 'Sobre'
menuItemSobre.addActionListener(this);
//Configura um evento ao se fechar a janela
this.addWindowListener(this);
//Adiciona o item de menu 'Novo' ao menu 'Arquivo'
menuArquivo.add(menuItemNovo);
//Adiciona o item de menu 'Sair' ao menu 'Arquivo'
menuArquivo.add(menuItemSair);
//Adiciona o menu 'Arquivo' à barra de menus
menu.add(menuArquivo);
```

```
        //Adiciona o item de menu 'Ajuda' ao menu 'Ajuda'
        menuAjuda.add(menuItemAjuda);
        //Adiciona o item de menu 'Sobre' ao menu 'Ajuda'
        menuAjuda.add(menuItemSobre);
        //Adiciona o menu 'Ajuda' à barra de menus
        menu.add(menuAjuda);
        //Retira a quebra de palavras na área de texto
        areaTexto.setLineWrap(true);
        //Retira a quebra de palavras na área de texto
        areaTexto.setWrapStyleWord(true);
        //Insere a área de texto em um painel deslizante
        painelDeslizante = new JScrollPane(areaTexto);

        painelDeslizante.setVerticalScrollBarPolicy(
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

        //Adiciona um rótulo vazio ao painel
        painelBotoes.add(new JLabel(""));
        //Adiciona o botão 'Imprimir' ao painel
        painelBotoes.add(botaoImprimir);
        //Adiciona um rótulo vazio ao painel
        painelBotoes.add(new JLabel(""));
        //Adiciona o botão 'Cancelar' ao painel
        painelBotoes.add(botaoCancelar);
        //Adiciona um rótulo vazio ao painel
        painelBotoes.add(new JLabel(""));
```

```
    //Adiciona a barra de menus ao painel superior
    painelSuperior.add(menu, BorderLayout.NORTH);
    //Adiciona o painel de botões ao painel superior
    painelSuperior.add(painelBotoes, BorderLayout.SOUTH);
    //Adiciona o painel superior ao painel principal
    painelPrincipal.add(painelSuperior, BorderLayout.NORTH);
    //Adiciona o painel deslizante ao painel principal
    painelPrincipal.add(painelDeslizante, BorderLayout.CENTER);
    //Configura o painel principal com padrão
    setContentPane(painelPrincipal);
    //Janela não redimensionável
    setResizable(false);
    //Janela visível
    setVisible(true);
    //Foco na área de texto
    areaTexto.requestFocus();
}

/*
 *    Método invocado quando ocorre um evento de fechamento de janela
 */
public void windowClosing(WindowEvent evento) {
    //Exibe uma mensagem confirmando a saída. Caso positivo, encerra o programa
```

```
        if(JOptionPane.showOptionDialog(this, "Deseja sair do programa?", "Sair", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, opcoesMensagem, opcoesMensagem[0]) == 0){
            System.exit(0);
        }
    }

    /*
     *  Método invocado quando ocorre um evento de fechamento de janela
     */
    public void windowClosed(WindowEvent evento) {
        //Exibe uma mensagem confirmando a saída. Caso positivo, encerra o programa
        if(JOptionPane.showOptionDialog(this, "Deseja sair do programa?", "Sair", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, opcoesMensagem, opcoesMensagem[0]) == 0){
            System.exit(0);
        }
    }

    /*
     *  Método sobrescrito da interface WindowListener
     */
    public void windowActivated(WindowEvent evento) {
    }

    /*
```

```
 *     Método sobrescrito da interface WindowListener
 */
public void windowDeactivated(WindowEvent evento) {
}

/*
 *     Método sobrescrito da interface WindowListener
 */
public void windowDeiconified(WindowEvent evento) {
}

/*
 *     Método sobrescrito da interface WindowListener
 */
public void windowIconified(WindowEvent evento) {
}

/*
 *     Método sobrescrito da interface WindowListener
 */
public void windowOpened(WindowEvent evento) {
}

/*
 *     Método invocado quando ocorre um evento de clique de botão ou item de menu
 */
```

```
public void actionPerformed(ActionEvent evento){

    //Caso o botão 'Imprimir' seja clicado
    if(evento.getSource() == botaoImprimir){
        //Caso a área de texto não esteja em branco
        if(!areaTexto.getText().equals("")){
            try{
                //Transcreve para Braille o texto inserido
                traducao = new TraducaoBraille(areaTexto.getText());
                //Instancia uma conexão com o Arduino
                conexaoSerial = new SerialTest();
                //Configura a transmissão ao Arduino e prossegue se não houver erros
                if(conexaoSerial.initialize(this, traducao) == true){
                    //Desabilita o menu 'Arquivo'
                    menuArquivoHabilitado(false);
                    //Desabilita o menu 'Ajuda'
                    menuAjudaHabilitado(false);
                    //Desabilita o botão 'Imprimir'
                    botaoImprimirHabilitado(false);
                    //Desabilita a edição da área de texto
                    areaTextoEditavel(false);
                    //Habilita o botão 'Cancelar'
                    botaoCancelarHabilitado(true);

                    //Transmite o texto transcrito ao Arduino
                    conexaoSerial.start();
```

```
                }

                //Caso algum caractere seja desconhecido, uma exceção é lançada e uma mensagem exibida
            } catch (ExcecaoCaractereDesconhecido e){
                JOptionPane.showMessageDialog(this, e.getMessage(), "Erro", JOptionPane.ERROR_MESSAGE);
            }
        }
        else
            //Caso a área de texto esteja em branco, exibe uma mensagem
            JOptionPane.showMessageDialog(this, "O texto est\u00e1 em branco", "Erro", JOptionPane.ERROR_MESSAGE);
        //Foco para a área de texto
        areaTexto.requestFocus();

    }else{
        //Caso o botão 'Cancelar' seja clicado
            if(evento.getSource() == botaoCancelar){
                //Exibe uma mensagem confirmando o cancelamento da impressão. Caso positivo, requisita o cancelamento
                if(JOptionPane.showOptionDialog(this, "Deseja cancelar a impress\u00e3o?", "Cancelar impress\u00e3o", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, opcoesMensagem, opcoesMensagem[0]) == 0){
                    //Desabilita o botão 'Cancelar'
                    botaoCancelarHabilitado(false);
                    //Requisita o cancelamento da impressão
```

```
                conexaoSerial.setCancelarImpressao(true);
            }
        }

        else{
            //Caso o item de menu 'Novo' seja clicado
            if(evento.getSource() == menuItemNovo){
                //Exibe uma mensagem confirmando um novo texto. Caso positivo, limpa a área de texto
                if(JOptionPane.showOptionDialog(this, "Deseja criar um novo texto?", "Novo texto", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, opcoesMensagem, opcoesMensagem[0]) == 0)
                    //Limpa a área de texto
                    areaTexto.setText("");
            }
            else{
                //Caso o item de menu 'Sair' seja clicado
                if(evento.getSource() == menuItemSair){
                    //Exibe uma mensagem confirmando a saída. Caso positivo, encerra o programa
                    if(JOptionPane.showOptionDialog(this, "Deseja sair do programa?", "Sair", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, opcoesMensagem, opcoesMensagem[0]) == 0)
                        System.exit(0);
                    }

                    else{
```

```
                    //Caso o item de menu 'Ajuda' seja clicado
                    if(evento.getSource() == menuItemAjuda){

                        //Configura a área de texto da ajuda
                        areaTextoAjuda.setLineWrap(true);
                        areaTextoAjuda.setWrapStyleWord(true);
                        areaTextoAjuda.setText(instrucoes);
                        areaTextoAjuda.setEditable(false);

                        //Exibe a ajuda
                         JOptionPane.showMessageDialog(this, areaTextoAjuda, "Ajuda do Impressora Braille", JOptionPane.PLAIN_MESSAGE);

                    }
                    else{
                        //Caso o item de menu 'Sobre' seja clicado
                        if(evento.getSource() == menuItemSobre){

                            //Exibe os créditos do programa
                             JOptionPane.showMessageDialog(this, "Programa Impressora Braille\n\nDesenvolvido como parte de Projeto Final\ndo curso de Engenharia de Computa\u00e7\u00e3o\ndo UniCEUB\n\nCriado por Gustavo Fontana Suzukawa\n ", "Sobre Impressora Braille", JOptionPane.PLAIN_MESSAGE);
                        }
                    }
                }
```

```
                }
            }
        }
    }
}

/*
 *    Método no estilo 'set' para habilitar ou desabilitar o menu 'Arquivo'
 */
public void menuArquivoHabilitado(boolean habilitado){
    menuArquivo.setEnabled(habilitado);
}

/*
 *    Método no estilo 'set' para habilitar ou desabilitar o menu 'Ajuda'
 */
public void menuAjudaHabilitado(boolean habilitado){
    menuAjuda.setEnabled(habilitado);
}

/*
 *    Método no estilo 'set' para habilitar ou desabilitar o botão 'Imprimir'
 */
public void botaoImprimirHabilitado(boolean habilitado){
    botaoImprimir.setEnabled(habilitado);
}
```

```
    /*
     *     Método no estilo 'set' para habilitar ou desabilitar o botão 'Cancelar'
     */
    public void botaoCancelarHabilitado(boolean habilitado){
        botaoCancelar.setEnabled(habilitado);
    }

    /*
     *     Método no estilo 'set' para habilitar ou desabilitar a área de texto
     */
    public void areaTextoEditavel(boolean habilitada){
        areaTexto.setEditable(habilitada);
    }

    /*
     *     Método main para criar um novo objeto da classe Impressora
     */
    public static void main(String args[]){
        Impressora impressora = new Impressora();
    }
}
```

2 – Classe SerialTest (adaptada)

```
/*
```

```
 *       Código adaptado do ARDUINO PLAYGROUND
 *       URL: http://www.arduino.cc/playground/Interfacing/Java
 */
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.util.Enumeration;
import javax.swing.*;
import gnu.io.PortInUseException;
import gnu.io.UnsupportedCommOperationException;
import java.util.TooManyListenersException;
/*
 *       A classe SerialTest se encarrega da comunicação com a placa Arduino.
 *       Repassa ao microcontrolador o texto transcrito para que esse seja
 *       impresso em Braille
 */

public class SerialTest extends Thread implements SerialPortEventListener{

    //Referência à classe SerialPort
    private SerialPort serialPort;
    //Nomes das portas usadas no Windows
```

```
    private static final String PORT_NAMES[] = { "COM4", "COM5" };
    //Stream de entrada
    private InputStream input;
    //Stream de saída
    private OutputStream output;
    //Tempo de espera pela abertura da porta
    private static final int TIME_OUT = 2000;
    //Taxa de transmissão em bits por segundo
    private static final int DATA_RATE = 115200;
    //Variáveis de comunicação e cancelamento de impressão
    private boolean liberado, cancelarImpressao;
    //Representação dos valores das agulhas na impressão
    private byte celula = 0, um = 1, dois = 2, quatro = 4, oito = 8, dezesseis = 16, trintaedois =
32, sessentaequatro = 64;

    //O objeto Impressora que chama a classe SerialTest
    private Impressora pai;
    //O objeto TraducaoBraille contendo o texto transcrito
    private TraducaoBraille traducaoBraille;

    /*
     *    Método invocado para configurar a transmissão com a
     *    placa Arduino e exibir mensagens sobre seu andamento
     */
    public boolean initialize(Impressora impressora, TraducaoBraille traducaoBraille){
```

```
//Configura o objeto Impressora
setPai(impressora);
//Configura o objeto TraducaoBraille
setTraducaoBraille(traducaoBraille);
//Configura uma variável de comunicação
setLiberado(false);
//Configura a variável de cancelamento de impressão
setCancelarImpressao(false);

//Identificador da porta
CommPortIdentifier portId = null;
//Portas disponíveis
Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

//Busca pela porta disponível
while (portEnum.hasMoreElements()) {
    CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.nextElement();
    for (String portName : PORT_NAMES) {
        if (currPortId.getName().equals(portName)) {
            portId = currPortId;
            break;
        }
    }
}

//Caso a porta não tenha sido encontrada
```

```
        if (portId == null) {
            //Exibe uma mensagem e retorna falso para não dar prosseguimento à impressão
            JOptionPane.showMessageDialog(getPai(), "Impressora n\u00e3o encontrada", "Erro", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        try {
            //Abre a porta serial
            serialPort = (SerialPort) portId.open(this.getClass().getName(), TIME_OUT);
            //Configura parâmetros
            serialPort.setSerialPortParams(DATA_RATE, SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
            //Abre as streams de entrada e saída
            input = serialPort.getInputStream();
            output = serialPort.getOutputStream();
            //Configura os eventos
            serialPort.addEventListener(this);
            serialPort.notifyOnDataAvailable(true);
            return true;
        }

        //Se alguma exceção for capturada, não dá prosseguimento à impressão
        catch (PortInUseException e) {
            return false;
        }
```

```
    catch (UnsupportedCommOperationException e) {
        return false;
    }
    catch (TooManyListenersException e) {
        return false;
    }
    catch (IOException e) {
        return false;
    }
}

/*
 *    Trata os eventos da porta serial
 */
public synchronized void serialEvent(SerialPortEvent oEvent) {
    //Se há dados disponíveis na porta
    if (oEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
        //Variável para receber mensagens do Arduino
        String mensagemRecebida = null;
        try {
            //Captura os dados da stream de entrada
            int available = input.available();
            //Ajusta os dados
            byte chunk[] = new byte[available];
            //Faz a leitura dos dados
            input.read(chunk, 0, available);
```

```
            //Formata a entrada
            mensagemRecebida = new String(chunk);

            /*
             *    Os dados recebidos sempre são mensagens "OK" vindas do Arduino.
             *    Qualquer exceção aqui é inútil, pois todo evento de dados
             *    disponíveis é conhecido.
             */
        } catch (IOException e) {
        } finally{
            if(mensagemRecebida.equals("OK"))
                //Libera o envio de novos dados ao Arduino
                setLiberado(true);
        }
    }
}

/*
 *    Encerra a conexão com o Arduino e reconfigura os componentes do objeto Impressora
 */
private synchronized void close() {
    //Se a porta serial não for nula
    if (serialPort != null) {
        //Remove a captura de eventos
        serialPort.removeEventListener();
        //Fecha a conexão
```

```
            serialPort.close();
            //Habilita o menu 'Arquivo'
            pai.menuArquivoHabilitado(true);
            //Habilita o menu 'Ajuda'
            pai.menuAjudaHabilitado(true);
            //Habilita o botão 'Imprimir'
            pai.botaoImprimirHabilitado(true);
            //Habilita a edição da área de texto
            pai.areaTextoEditavel(true);
            //Desabilita o botão 'Cancelar'
            pai.botaoCancelarHabilitado(false);
            //Ajusta a variável de cancelamento de impressão
            setCancelarImpressao(false);
        }
    }

    /*
     *     Por estender de Thread, sobrescreve o método run
     */
    public void run(){
        try{
            //Se o envio do texto transcrito ao microcontrolador foi correto
            if(enviaTexto() == true)
                //Exibe uma mensagem confirmando a impressão do texto
                JOptionPane.showMessageDialog(getPai(), "Texto impresso", "Impress\u00e3o",
JOptionPane.INFORMATION_MESSAGE);
```

```
            else
                //Exibe uma mensagem informando o erro na impressão
                JOptionPane.showMessageDialog(getPai(), "Houve um erro na impress\u00e3o", "Erro", JOptionPane.ERROR_MESSAGE);

            //Caso a impressão deva ser cancelada
        } catch(ExcecaoImpressaoCancelada e){
            //Exibe uma mensagem confirmando o cancelamento da impressão
            JOptionPane.showMessageDialog(getPai(), e.getMessage(), "Impress\u00e3o", JOptionPane.INFORMATION_MESSAGE);
        }finally{
            //Independente do resultado, libera os recursos alocados
            close();
        }
    }

    /*
     *  Envia o texto transcrito ao Arduino
     */
    private boolean enviaTexto() throws ExcecaoImpressaoCancelada{
        int contadorBraille = 0;
        EstruturaAgulhas textoBraille[] = traducaoBraille.getTextoBraille();
        try{
            //Envia célula por célula
            while(textoBraille[contadorBraille].getUtilizado() != false){
```

```
            //A cada 120 células enviadas, espera a confirmação do Arduino
            //de que imprimiu todas, devido à limitação de 120 bytes de seu buffer
            if(contadorBraille % 120 == 0){
                //Chama o método que aguarda a resposta
                aguardaConfirmacao();
                //Configura a variável de comunicação
                setLiberado(false);
            }

            //Chama o método que envia a célula
            enviaSinais(textoBraille[contadorBraille]);
            contadorBraille++;
        }

        //Chama o método que aguarda a confirmação da impressão
        aguardaConfirmacao();
        return true;

        //Se houver uma exceção cancela a impressão
    } catch (IOException e) {
        return false;
    }
}

/*
 *     Envia célula para ser impressa
```

```
 */
private void enviaSinais(EstruturaAgulhas estruturaBraille) throws IOException{
    //Lê cada flag da célula
    if(estruturaBraille.getOutrasFuncoes() == true)
        celula += um;
    if(estruturaBraille.getUm() == true)
        celula += dois;
    if(estruturaBraille.getDois() == true)
        celula += quatro;
    if(estruturaBraille.getTres() == true)
        celula += oito;
    if(estruturaBraille.getQuatro() == true)
        celula += dezesseis;
    if(estruturaBraille.getCinco() == true)
        celula += trintaedois;
    if(estruturaBraille.getSeis() == true)
        celula += sessentaequatro;

    //Envia a célula
    output.write(celula);
    celula = 0;
}

/*
 *     Aguarda resposta do microcontrolador
 */
```

```
private void aguardaConfirmacao() throws ExcecaoImpressaoCancelada{
    try{
        //Enquanto não houver resposta
        while(getLiberado() == false){
            //Faz a thread dormir por três segundos
            Thread.sleep(3000);
        }

        //Configura a variável de comunicação
        setLiberado(false);

        //Caso tenha sido ordenado o cancelamento da impressão
        if(getCancelarImpressao() == true){
            //Lança uma nova exceção de cancelamento
            throw new ExcecaoImpressaoCancelada();
        }

    //Interrupção da thread
    } catch(InterruptedException e){
        //Lança uma nova exceção de cancelamento
        throw new ExcecaoImpressaoCancelada();
    }
}

/*
 *     Retorna a variável de comunicação
```

```
 */
public boolean getLiberado(){
    return liberado;
}

/*
 *     Seta a variável de comunicação
 */
public void setLiberado(boolean liberado){
    this.liberado = liberado;
}

/*
 *     Retorna a variável de cancelamento de impressão
 */
public boolean getCancelarImpressao(){
    return cancelarImpressao;
}

/*
 *     Seta a variável de cancelamento de impressão
 */
public void setCancelarImpressao(boolean cancelarImpressao){
    this.cancelarImpressao = cancelarImpressao;
}
```

```
/*
 *     Retorna o objeto Impressora
 */
public Impressora getPai(){
    return pai;
}

/*
 *     Seta o objeto Impressora
 */
public void setPai(Impressora pai){
    this.pai = pai;
}

/*
 *     Retorna o objeto TraducaoBraille
 */
public TraducaoBraille getTraducaoBraille(){
    return traducaoBraille;
}

/*
 *     Seta o objeto TraducaoBraille
 */
public void setTraducaoBraille(TraducaoBraille traducaoBraille){
        this.traducaoBraille = traducaoBraille;
```

```
    }
}
```

3 – Classe EstruturaAgulhas

```
/*
 *      Autor: Gustavo Fontana Suzukawa
 *      Implementa a estrutura das agulhas de uma célula Braille
 */

public class EstruturaAgulhas{
        //Variáveis representando as agulhas
        private boolean um, dois, tres, quatro, cinco, seis, outrasFuncoes, utilizado;
        /*
         *      Construtor configurando as variáveis
        */
        public EstruturaAgulhas(){
                //Configura as variáveis
                um = false;
                dois = false;
                tres = false;
                quatro = false;
                cinco = false;
                seis = false;
                outrasFuncoes = false;
                utilizado = false;
```

```
}

/*
 *	Retorna a variável do ponto 1 da agulha
 */
public boolean getUm(){
	return um;
}

/*
 *	Seta a variável do ponto 1 da agulha
 */
public void setUm(boolean novoUm){
	this.um = novoUm;
}

/*
 *	Retorna a variável do ponto 2 da agulha
 */
public boolean getDois(){
	return dois;
}

/*
 *	Seta a variável do ponto 2 da agulha
 */
```

```
public void setDois(boolean novoDois){
	this.dois = novoDois;
}

/*
 *	Retorna a variável do ponto 3 da agulha
 */
public boolean getTres(){
	return tres;
}

/*
 *	Seta a variável do ponto 3 da agulha
 */
public void setTres(boolean novoTres){
	this.tres = novoTres;
}

/*
 *	Retorna a variável do ponto 4 da agulha
 */
public boolean getQuatro(){
	return quatro;
}

/*
```

```
 *      Seta a variável do ponto 4 da agulha
 */
public void setQuatro(boolean novoQuatro){
        this.quatro = novoQuatro;
}

/*
 *      Retorna a variável do ponto 5 da agulha
 */
public boolean getCinco(){
        return cinco;
}

/*
 *      Seta a variável do ponto 5 da agulha
 */
public void setCinco(boolean novoCinco){
        this.cinco = novoCinco;
}

/*
 *      Retorna a variável do ponto 6 da agulha
 */
public boolean getSeis(){
        return seis;
}
```

```
/*
 *	Seta a variável do ponto 6 da agulha
 */
public void setSeis(boolean novoSeis){
	this.seis = novoSeis;
}

/*
 *	Retorna a variável de outras funções
 */
public boolean getOutrasFuncoes(){
	return outrasFuncoes;
}

/*
 *	Seta a variável de outras funções
 */
public void setOutrasFuncoes(boolean outrasFuncoes){
	this.outrasFuncoes = outrasFuncoes;
}

/*
 *	Retorna a variável de célula utilizada
 */
public boolean getUtilizado(){
```

```
		return utilizado;
	}

	/*
	 *	Seta a variável de célula utilizada
	 */
	public void setUtilizado(boolean novoUtilizado){
		this.utilizado = novoUtilizado;
	}
}
```

4 – Classe TraducaoBraille

```
/*
 *	Autor: Gustavo Fontana Suzukawa
 *	Classe para transcrição de textos
 */

public class TraducaoBraille {
    //Variáveis de controle
    private int contadorLegivel;
    private int contadorBraille;
    //O texto transcrito é armazenado nesta variável
    private EstruturaAgulhas textoBraille[];

    /*
```

```
 *    Construtor da classe
 */
public TraducaoBraille(String textoLegivel) throws ExcecaoCaractereDesconhecido{
    //Configura as variáveis
    contadorLegivel = 0;
    contadorBraille = 0;
    textoBraille = new EstruturaAgulhas[textoLegivel.length() * 5];
    //Invoca a transcrição do texto para Braille
    traduzTexto(textoLegivel);
}

/*
 *    Realiza a transcrição do texto para Braille
 */
private void traduzTexto(String textoLegivel) throws ExcecaoCaractereDesconhecido{

    //Cada caractere é armazenado nessa estrutura
    EstruturaAgulhas estruturaAtual;
    //Variável para controle das palavras maiúsculas
    int contadorMaiuscula = 0;

    //Devido a um problema em se comparar um apóstrofo, ele é armazenado em um string
    String apostrofo = "'";
    //Variável para controle de números ordinais
    boolean numeroOrdinal = false;
```

```
        //Para cada caractere do texto, uma transcrição diferente
        for(contadorLegivel = 0; contadorLegivel < textoLegivel.length(); contadorLegivel++){

            //Inicializa cada estrutura
            textoBraille[contadorBraille] = new EstruturaAgulhas();

            //Se for uma letra maiúscula
            if(letraMaiuscula(textoLegivel.charAt(contadorLegivel)) == true){

                //Substitui a letra por uma equivalente minúscula
                textoLegivel = textoLegivel.substring(0, contadorLegivel) +
Character.toLowerCase(textoLegivel.charAt(contadorLegivel)) +
textoLegivel.substring(contadorLegivel + 1, textoLegivel.length());

                //Incrementa o contador específico
                contadorMaiuscula++;

            //Se for uma letra minúscula
            }else{
                //Se antes dessa letra havia uma maiúscula
                if(contadorMaiuscula != 0){

                    //Configura a célula de letra maiúscula
                    estruturaAtual = new EstruturaAgulhas();
                    estruturaAtual.setQuatro(true);
                    estruturaAtual.setSeis(true);
```

```
                estruturaAtual.setUtilizado(true);

                //Insere essa célula no texto transcrito
                insereNoTextoBraille(estruturaAtual, contadorBraille - contadorMaiuscula, contadorBraille);
                contadorBraille++;

                //Se toda a palavra anterior for maiúscula, insere outra célula igual
                if(contadorMaiuscula > 1){
                    insereNoTextoBraille(estruturaAtual, contadorBraille - contadorMaiuscula, contadorBraille);
                    contadorBraille++;
                }
                contadorMaiuscula = 0;
            }
        }

        //Se o caractere for um dígito
        if(Character.isDigit(textoLegivel.charAt(contadorLegivel)) == true){

            //Verifica se trata-se de um número ordinal
            numeroOrdinal = verificaOrdinal(textoLegivel, contadorLegivel);

            //Se for o primeiro caractere, configura e insere a célula de representação de número
            if(contadorLegivel == 0){
```

```
            textoBraille[contadorBraille].setTres(true);
            textoBraille[contadorBraille].setQuatro(true);
            textoBraille[contadorBraille].setCinco(true);
            textoBraille[contadorBraille].setSeis(true);
            textoBraille[contadorBraille].setUtilizado(true);
            contadorBraille++;
            textoBraille[contadorBraille] = new EstruturaAgulhas();
        }
        else{
            //Faz o mesmo se o caractere anterior não é um dígito
            if(Character.isDigit(textoLegivel.charAt(contadorLegivel - 1)) == false){

                textoBraille[contadorBraille].setTres(true);
                textoBraille[contadorBraille].setQuatro(true);
                textoBraille[contadorBraille].setCinco(true);
                textoBraille[contadorBraille].setSeis(true);
                textoBraille[contadorBraille].setUtilizado(true);
                contadorBraille++;
                textoBraille[contadorBraille] = new EstruturaAgulhas();
            }
        }
    }

    //Compara se trata-se de um apóstrofo
    if(textoLegivel.substring(contadorLegivel, contadorLegivel + 1).equals(apostrofo)){
    //' (apóstrofo)
```

```
    textoBraille[contadorBraille].setTres(true);
}

/*
 *      Para os demais casos uma ampla gama de possibilidades.
 *      Para cada um deles, configura-se as agulhas que devem
 *      ser acionadas. Algumas situações demandam mais de uma
 *      célula
 */
switch(textoLegivel.charAt(contadorLegivel)){

   case 'a':                    //letra minúscula a
        textoBraille[contadorBraille].setUm(true);
        break;

   case 'b':                    //letra minúscula b
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        break;

   case 'c':                    //letra minúscula c
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

   case 'd':                    //letra minúscula d
```

```
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case 'e':                   //letra minúscula e
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case 'f':                           //letra minúscula f
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

    case 'g':                   //letra minúscula g
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case 'h':                   //letra minúscula h
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
```

```
        textoBraille[contadorBraille].setCinco(true);
        break;

    case 'i':                           //letra minúscula i
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

    case 'j':                           //letra minúscula j
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case 'k':                   //letra minúscula k
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        break;

    case 'l':                           //letra minúscula l
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        break;

    case 'm':                   //letra minúscula m
```

```
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

case 'n':                       //letra minúscula n
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case 'o':                       //letra minúscula o
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case 'p':                       //letra minúscula p
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

case 'q':                       //letra minúscula q
```

```
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case 'r':                               //letra minúscula r
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case 's':                           //letra minúscula s
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

case 't':                               //letra minúscula t
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;
```

```
case 'u':                    //letra minúscula u
    textoBraille[contadorBraille].setUm(true);
    textoBraille[contadorBraille].setTres(true);
    textoBraille[contadorBraille].setSeis(true);
    break;

case 'v':                    //letra minúscula v
    textoBraille[contadorBraille].setUm(true);
    textoBraille[contadorBraille].setDois(true);
    textoBraille[contadorBraille].setTres(true);
    textoBraille[contadorBraille].setSeis(true);
    break;

case 'w':                    //letra minúscula w
    textoBraille[contadorBraille].setDois(true);
    textoBraille[contadorBraille].setQuatro(true);
    textoBraille[contadorBraille].setCinco(true);
    textoBraille[contadorBraille].setSeis(true);
    break;

case 'x':                    //letra minúscula x
    textoBraille[contadorBraille].setUm(true);
    textoBraille[contadorBraille].setTres(true);
    textoBraille[contadorBraille].setQuatro(true);
    textoBraille[contadorBraille].setSeis(true);
```

```
        break;

case 'y':                       //letra minúscula y
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case 'z':                       //letra minúscula z
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00e7':                  //ç (letra c com cedilha)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00e1':                  //á (letra a com acento agudo)
```

```
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00e9':                  //é (letra e com acento agudo)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00ed':                  //í (letra i com acento agudo)
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

case '\u00f3':                  //ó (letra o com acento agudo)
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        break;
```

```
case '\u00fa':                  //ú (letra u com acento agudo)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00e0':                  //à (letra a com acento grave)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00e2':                  //â (letra a com acento circunflexo)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00ea':                  //ê (letra e com acento circunflexo)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setSeis(true);
```

```
        break;

case '\u00f4':                  //ô (letra o com acento circunflexo)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00e3':                  //ã (letra a com til)
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case '\u00f5':                  //õ (letra o com til)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00fc':                  //ü (letra u com trema)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
```

```
        break;

case '\u002c':              //, (vírgula)
        textoBraille[contadorBraille].setDois(true);
        break;

case '\u003b':              //; (ponto-e-vírgula)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        break;

case '\u003a':              //: (dois pontos)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case '\u002e':              //. (ponto final)
        textoBraille[contadorBraille].setTres(true);
        break;

case '\u003f':              //? (ponto de interrogação)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u0021':              //! (ponto de exclamação)
```

```
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case '\u002d':                  //- (hífen)
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u2014':                  //— (travessão)
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u2022':                  //• (marcador ou círculo)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
```

```
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case '\u002a':                  //* (asterisco)
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case '\u0026':                  //& (E comercial)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u002f':                  /// (barra)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        break;
```

```
case '\u007c':                  //| (barra vertical)
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u2192':                  //seta à direita
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case '\u2190':                  //seta à esquerda
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
```

```
        textoBraille[contadorBraille].setCinco(true);
        break;

    case '\u2194':                  //seta de duplo sentido
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case '\u20ac':                  //€ (euro)
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setUm(true);
```

```
        textoBraille[contadorBraille].setCinco(true);
        break;

    case '\u0024':              //$ (cifrão)
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u0025':              //% (por cento)
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u2030':              //‰ (por mil)
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
```

```
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u00a7':                  //§ (parágrafo jurídico)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        break;

    case '\u002b':                  //+  (sinal de adição)
        textoBraille[contadorBraille].setDois(true);
```

```
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case '\u00f7':              //÷ (sinal de divisão)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u003d':              //= (sinal de igualdade)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u003e':              //> (sinal de maior que)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case '\u003c':              //< (sinal de menor que)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setQuatro(true);
```

```
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00b0':              //° (grau)
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u0020':              //  (espaçamento)
        break;

case '\u0031':              //1 (dígito 1)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
        }
        break;

case '\u0032':              //2 (dígito 2)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
                textoBraille[contadorBraille].setDois(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
```

```
                textoBraille[contadorBraille].setTres(true);
        }
        break;

case '\u0033':                  //3 (dígito 3)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
                textoBraille[contadorBraille].setQuatro(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setCinco(true);
        }
        break;

case '\u0034':                  //4 (dígito 4)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
                textoBraille[contadorBraille].setQuatro(true);
                textoBraille[contadorBraille].setCinco(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setCinco(true);
                textoBraille[contadorBraille].setSeis(true);
        }
        break;
```

```
case '\u0035':                  //5 (dígito 5)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
                textoBraille[contadorBraille].setCinco(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setSeis(true);
        }
        break;

case '\u0036':                  //6 (dígito 6)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setQuatro(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setTres(true);
                textoBraille[contadorBraille].setCinco(true);
        }
        break;

case '\u0037':                  //7 (dígito 7)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
                textoBraille[contadorBraille].setDois(true);
```

```
                textoBraille[contadorBraille].setQuatro(true);
                textoBraille[contadorBraille].setCinco(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setTres(true);
                textoBraille[contadorBraille].setCinco(true);
                textoBraille[contadorBraille].setSeis(true);
        }
        break;

case '\u0038':                  //8 (dígito 8)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setUm(true);
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setCinco(true);
        }else{
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setTres(true);
                textoBraille[contadorBraille].setSeis(true);
        }
        break;

case '\u0039':                  //9 (dígito 9)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setQuatro(true);
```

```
        }else{
                textoBraille[contadorBraille].setTres(true);
                textoBraille[contadorBraille].setCinco(true);
        }
        break;

case '\u0030':                  //0 (dígito 0)
        if(numeroOrdinal == false){
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setQuatro(true);
                textoBraille[contadorBraille].setCinco(true);
        }else{
                textoBraille[contadorBraille].setTres(true);
                textoBraille[contadorBraille].setCinco(true);
                textoBraille[contadorBraille].setSeis(true);
        }
        break;

case '\u0022':                  //" (aspas duplas)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u201c':                  //“ (aspas duplas à direita)
        textoBraille[contadorBraille].setDois(true);
```

```
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u201d':              // aspas duplas à esquerda
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u201e':              //„ (aspas duplas à direita)
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u00ab':              //« (aspas angulares à esquerda)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;
```

```
case '\u00bb':                  //» (aspas angulares à direita)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u2039':                  //‹ (aspas simples angulares à esquerda)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u203a':                  //› (aspas simples angulares à direita)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
```

```
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u2018':              //‘ (aspas simples à esquerda)
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

    case '\u2019':              //’ (aspas simples à direita)
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setSeis(true);
        break;
```

```
case '\u0028':                  //( (parêntese esquerdo)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        break;

case '\u0029':                  //) (parêntese direito)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

case '\ufd3e':                  //﴾ (parêntese esquerdo ornamentado)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
```

```
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        break;

    case '\ufd3f':                  //) (parêntese direito ornamentado)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        break;

    case '\u005b':                  //[ (colchete à esquerda)
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setTres(true);
        break;
```

```
case '\u005d':                  //] (colchete à direita)
        textoBraille[contadorBraille].setSeis(true);
        textoBraille[contadorBraille].setUtilizado(true);
        contadorBraille++;
        textoBraille[contadorBraille] = new EstruturaAgulhas();
        textoBraille[contadorBraille].setDois(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setQuatro(true);
        textoBraille[contadorBraille].setCinco(true);
        textoBraille[contadorBraille].setSeis(true);
        break;

case '\u00aa':                  //ª (ordinal feminino)
        if(numeroOrdinal == true)
                numeroOrdinal = false;
        textoBraille[contadorBraille].setUm(true);
        break;

case '\u00ba':                  //º (ordinal masculino)
        if(numeroOrdinal == true)
                numeroOrdinal = false;
        textoBraille[contadorBraille].setUm(true);
        textoBraille[contadorBraille].setTres(true);
        textoBraille[contadorBraille].setCinco(true);
        break;
```

```
            case '\u0009':                  //TAB (tabulação)
                for(byte contadorTabulacao = 1; contadorTabulacao < 4; contadorTabulacao++){
                    textoBraille[contadorBraille].setUtilizado(true);
                    contadorBraille++;
                    textoBraille[contadorBraille] = new EstruturaAgulhas();
                }
                break;

            case '\n':                      //quebra de linha
                textoBraille[contadorBraille].setTres(true);
                textoBraille[contadorBraille].setOutrasFuncoes(true);
                textoBraille[contadorBraille].setUtilizado(true);
                contadorBraille++;
                textoBraille[contadorBraille] = new EstruturaAgulhas();
                textoBraille[contadorBraille].setDois(true);
                textoBraille[contadorBraille].setOutrasFuncoes(true);
                break;

            default:                        //caractere não conhecido
                if(!textoLegivel.substring(contadorLegivel, contadorLegivel + 1).equals(apostrofo))
                    //Caso um caractere seja desconhecido, é lançada uma exceção
                    throw new ExcecaoCaractereDesconhecido(textoLegivel.charAt(contadorLegivel));
        }
```

```
            textoBraille[contadorBraille].setUtilizado(true);
            contadorBraille++;
        }

        //Mesma verificação do início
        if(contadorMaiuscula != 0){

            estruturaAtual = new EstruturaAgulhas();
            estruturaAtual.setQuatro(true);
            estruturaAtual.setSeis(true);
            estruturaAtual.setUtilizado(true);

            insereNoTextoBraille(estruturaAtual, contadorBraille - contadorMaiuscula,
contadorBraille);
            contadorBraille++;

            if(contadorMaiuscula > 1){
                insereNoTextoBraille(estruturaAtual, contadorBraille - contadorMaiuscula,
contadorBraille);
                contadorBraille++;
            }

            contadorMaiuscula = 0;
        }
```

```
    //No final acrescenta-se uma célula indicando final da mensagem
    textoBraille[contadorBraille] = new EstruturaAgulhas();
    textoBraille[contadorBraille].setUtilizado(true);
    textoBraille[contadorBraille].setUm(true);
    textoBraille[contadorBraille].setOutrasFuncoes(true);
    contadorBraille++;

    textoBraille[contadorBraille] = new EstruturaAgulhas();
    textoBraille[contadorBraille].setUtilizado(false);

    //Invoca a inserção de quebras de linha e de página
    formataTexto();
}

/ *
  *   Formata o texto já transcrito inserindo quebras de linha e de página.
  *   Cada linha possui no máximo 30 caracteres.
  *   Cada página possui no máximo 27 linhas.
  * /
private void formataTexto(){
    //Variáveis de controle
    int contadorCaracteres = 0, contadorLinhas = 1, ultimaEstrutura = contadorBraille;
    EstruturaAgulhas estruturaAtual;
    contadorBraille = 0;

    //Enquanto houver células
```

```
        while(textoBraille[contadorBraille].getUtilizado() != false){
            contadorCaracteres++;
            //Caso haja uma quebra de linha já inserida no texto
            if(textoBraille[contadorBraille].getOutrasFuncoes() == true &&
textoBraille[contadorBraille].getTres() == true){
                //Configuras as variáveis
                contadorCaracteres = 0;
                contadorLinhas++;
                contadorBraille++;
            }

            //Se existem 30 caracteres seguidos sem quebra de linha
            if(contadorCaracteres == 31){
                //Verifica onde se deve quebrar a linha
                if(quebraDePalavra(contadorBraille) != -1){
                    contadorBraille = quebraDePalavra(contadorBraille);
                }

                //Configura e insere uma quebra de página
                estruturaAtual = new EstruturaAgulhas();
                estruturaAtual.setTres(true);
                estruturaAtual.setOutrasFuncoes(true);
                estruturaAtual.setUtilizado(true);
                insereNoTextoBraille(estruturaAtual, contadorBraille, ultimaEstrutura);
                ultimaEstrutura++;
                contadorBraille++;
```

```
            estruturaAtual = new EstruturaAgulhas();
            estruturaAtual.setDois(true);
            estruturaAtual.setOutrasFuncoes(true);
            estruturaAtual.setUtilizado(true);
            insereNoTextoBraille(estruturaAtual, contadorBraille, ultimaEstrutura);
            ultimaEstrutura++;
            contadorLinhas++;
            contadorCaracteres = 0;
        }

        //Se existem 27 linhas
        if(contadorLinhas == 28){

            //Insere uma quebra de página
            textoBraille[contadorBraille - 1].setTres(false);
            textoBraille[contadorBraille - 1].setQuatro(true);

            contadorLinhas = 1;
        }

        contadorBraille++;
    }
}

/ *
```

```
    *   Verifica se uma palavra será quebrada pela quebra de linha
    * /
   private int quebraDePalavra(int contador){

      //Procura por espaços nos caracteres antes e depois da quebra de linha
      if(verificaSeEspaco(textoBraille[contador]) == false &&
verificaSeEspaco(textoBraille[contador - 1]) == false){
         //Procura a posição no texto transcrito anterior à palavra quebrada
         while(verificaSeEspaco(textoBraille[contador]) == false){
            if((textoBraille[contador].getOutrasFuncoes() == true &&
textoBraille[contador].getDois() == true) || contador == 0)
               return -1;
            contador--;
         }

         //Retorna a posição correta de quebra de linha
         return contador + 1;
      }
      //Se nenhuma palavra for quebrada
      else
         return -1;
   }

   / *
    *   Verifica se a célula é um caractere de espaço
    * /
```

```
private boolean verificaSeEspaco(EstruturaAgulhas estrutura){

    //Simplesmente verifica se a estrutura abriga uma célula contendo um espaçamento
    if(estrutura.getOutrasFuncoes() == false && estrutura.getUm() == false &&
                        estrutura.getDois() == false && estrutura.getTres() == false &&
                        estrutura.getQuatro() == false && estrutura.getCinco() == false &&
                        estrutura.getSeis() == false)
        return true;

    else
        return false;
}

/ *
  *    Verifica se o caractere é uma letra maiúscula
  */
private boolean letraMaiuscula(char letra){

    /*
     * Verifica se o caractere é uma letra maiúscula,
     * levando em conta todas as possibilidades do alfabeto
     */
    if(letra == '\u0041' || letra == '\u0042' || letra == '\u0043' || letra == '\u0044' ||
                    letra == '\u0045' || letra == '\u0046' || letra == '\u0047' || letra == '\u0048' ||
                    letra == '\u0049' || letra == '\u004a' || letra == '\u004b' || letra == '\u004c' ||
                    letra == '\u004d' || letra == '\u004e' || letra == '\u004f' || letra == '\u0050' ||
```

```
                letra == '\u0051' || letra == '\u0052' || letra == '\u0053' || letra == '\u0054' ||
                letra == '\u0055' || letra == '\u0056' || letra == '\u0057' || letra == '\u0058' ||
                letra == '\u0059' || letra == '\u005a' || letra == '\u00c7' || letra == '\u00c1' ||
                letra == '\u00c9' || letra == '\u00cd' || letra == '\u00d3' || letra == '\u00da' ||
                letra == '\u00c0' || letra == '\u00c2' || letra == '\u00ca' || letra == '\u00d4' ||
                letra == '\u00c3' || letra == '\u00d5' || letra == '\u00dc')
            return true;
        else
            return false;
    }

    / *
     *    Insere uma célula no texto transcrito
     * /
    private void insereNoTextoBraille(EstruturaAgulhas estrutura, int posicaoInserir, int
posicaoFinal){

        int posicaoAtual = -1;
        while(posicaoFinal - posicaoAtual - 1 >= posicaoInserir){
            //Empurra as células uma posição acima
            textoBraille[posicaoFinal - posicaoAtual] = textoBraille[posicaoFinal - posicaoAtual -
1];
            posicaoAtual++;
        }

        //Insere a célula na posição desejada
```

```
        textoBraille[posicaoInserir] = estrutura;
    }

    / *
     *    Verifica se uma porção de texto é um número ordinal
     * /
    private boolean verificaOrdinal(String textoLegivel, int contador){

        //Enquanto o caractere for um dígito
        while(Character.isDigit(textoLegivel.charAt(contador)) == true){
            if(contador == textoLegivel.length() - 1)
                return false;
            contador++;
        }

        //Se for um número ordinal
        if(textoLegivel.charAt(contador) == '\u00aa' || textoLegivel.charAt(contador) ==
'\u00ba')
            return true;
        else
            return false;
    }

    / *
     *    Retorna o texto transcrito
     * /
```

```
    public EstruturaAgulhas[] getTextoBraille(){
        return textoBraille;
    }
}
```

5 – Classe ExcecaoCaractereDesconhecido

```
/*
 *      Autor: Gustavo Fontana Suzukawa
 */
import java.lang.Exception;
/*
 *      Exceção para caracteres desconhecidos durante o
 *      processo de transcrição de textos.
 */

public class ExcecaoCaractereDesconhecido extends Exception{
        /*
         *      Construtor com o caractere desconhecido
         */
        public ExcecaoCaractereDesconhecido(char caractereDesconhecido){
                super("Caractere inv\u00e1lido: " + caractereDesconhecido);
        }
}
```

6 – Classe ExcecaoImpressaoCancelada

```
/*
 *      Autor: Gustavo Fontana Suzukawa
 */
import java.lang.Exception;
/*
 *      Exceção para cancelar a impressão sendo
 *      executada no momento.
 */

public class ExcecaoImpressaoCancelada extends Exception{
        /*
         *      Construtor da exceção
         */
        public ExcecaoImpressaoCancelada(){
                super("Impress\u00e3o cancelada");
        }
}
```

APÊNDICE B – Código presente no microcontrolador ATmega2560

```
//Arquivo final.pde
//Código presente no microcontrolador ATmega2560

//Define as portas das agulhas e demais funções
#define pinoRolete1 46
#define pinoRolete2 47
#define pinoRolete3 48
#define pinoRolete4 49
#define pinoCarro1 50
#define pinoCarro2 51
#define pinoCarro3 52
#define pinoCarro4 53
#define agulha1 22
#define agulha2 23
#define agulha3 24
#define agulha4 25
#define agulha5 26
#define agulha6 27

//Variáveis de controle
byte quociente, contador;
int contadorSerial, cabecaEsquerda = 0;
```

```
//Função setup para configurar as portas e a transmissão
void setup(){

    //Configura as portas para saída
    pinMode(pinoRolete1, OUTPUT);
    pinMode(pinoRolete2, OUTPUT);
    pinMode(pinoRolete3, OUTPUT);
    pinMode(pinoRolete4, OUTPUT);
    pinMode(pinoCarro1, OUTPUT);
    pinMode(pinoCarro2, OUTPUT);
    pinMode(pinoCarro3, OUTPUT);
    pinMode(pinoCarro4, OUTPUT);
    pinMode(agulha1, OUTPUT);
    pinMode(agulha2, OUTPUT);
    pinMode(agulha3, OUTPUT);
    pinMode(agulha4, OUTPUT);
    pinMode(agulha5, OUTPUT);
    pinMode(agulha6, OUTPUT);

    //Quebra de página inicial do programa
    quebraDePaginaInicial();

    //Configura a transmissão
    Serial.begin(115200);
    delay(2000);
```

```
    //Avisa o computador que está pronto
    Serial.print("OK");

    contadorSerial = 0;

}

//Função loop que é executada ciclicamente
void loop(){

    //Desliga as funções a cada passagem
    desligaFuncoes();

    //Se algum caractere foi impresso
    if(cabecaEsquerda != 0 && quociente != 9 && quociente != 17 && quociente != 5 && quociente != 3 && Serial.available() > 0){

        //Aciona o movimento de cabeça de impressão para a direita
        cabecaParaDireita();

    }

    //Se houver um dado no buffer da porta
    if(Serial.available()){

        //Lê os dados do buffer
```

```
quociente = Serial.read();

//Se trata-se de um caractere a ser impresso
if(quociente % 2 == 0){

   quociente = quociente / 2;

   //Lê o byte recebido
   for(contador = 1; contador <= 6 && quociente != 0; contador++){

      //Verifica quais agulhas devem ser acionadas
      if(quociente % 2 == 1)
         digitalWrite(28 - contador, HIGH);

      quociente = quociente / 2;

   }

   //Aguarda as agulhas pressionarem o papel e depois elas são desativadas
   delay(200);
   digitalWrite(agulha1, LOW);
   digitalWrite(agulha2, LOW);
   digitalWrite(agulha3, LOW);
   digitalWrite(agulha4, LOW);
   digitalWrite(agulha5, LOW);
   digitalWrite(agulha6, LOW);
```

```
    delay(400);

    cabecaEsquerda++;

}
//Se trata-se de uma funcionalidade da impressora
else{

    switch(quociente){

        //Quebra de linha
        case 9:
            delay(200);
            quebraDeLinha();
            delay(200);
            break;

        //Quebra de página
        case 17:
            delay(200);
            quebraDePaginaFinal();
            delay(20000);
            quebraDePaginaInicial();
            delay(200);
            break;
```

```
        //Movimento da cabeça de impressão para esquerda
        case 5:
            delay(200);
            cabecaParaEsquerda(cabecaEsquerda);
            delay(200);
            cabecaEsquerda = 0;
            break;

        //Fim de texto
        case 3:
            //Confirma o fim do texto
            delay(1000);
            cabecaParaEsquerda(cabecaEsquerda);
            delay(200);
            cabecaEsquerda = 0;
            quebraDePaginaFinal();
            delay(1000);
            Serial.print("OK");
            break;

      }

   }
   contadorSerial++;

}
```

```
    //Se não houver dados no buffer
    else{

        //Se foram lidas 120 estruturas
        if(contadorSerial == 120){

            //Limpa o buffer da porta
            Serial.flush();
            contadorSerial = 0;

            //Requisita mais dados
            Serial.print("OK");

        }

    }

}

//Função para a quebra de linha
void quebraDeLinha(){
    quebra(20);
}

//Função para a quebra de página inicial
void quebraDePaginaInicial(){
```

```
    quebra(100);
}

//Função para a quebra de página final
void quebraDePaginaFinal(){
    quebra(540);
}

//Função para genérica para a quebra
void quebra(int espacamento){

    for(int counter = 0; counter < espacamento; counter++){
        digitalWrite(pinoRolete1, HIGH);
        delay(10);
        digitalWrite(pinoRolete1, LOW);
        digitalWrite(pinoRolete2, HIGH);
        delay(10);
        digitalWrite(pinoRolete2, LOW);
        digitalWrite(pinoRolete3, HIGH);
        delay(10);
        digitalWrite(pinoRolete3, LOW);
        digitalWrite(pinoRolete4, HIGH);
        delay(10);
        digitalWrite(pinoRolete4, LOW);
    }
```

```
}

//Função para movimentar a cabeça de impressão para a direita
void cabecaParaDireita(){

    for(int counter = 0; counter < 4; counter++){
        digitalWrite(pinoCarro1, HIGH);
        delay(5);
        digitalWrite(pinoCarro1, LOW);
        digitalWrite(pinoCarro2, HIGH);
        delay(5);
        digitalWrite(pinoCarro2, LOW);
        digitalWrite(pinoCarro3, HIGH);
        delay(5);
        digitalWrite(pinoCarro3, LOW);
        digitalWrite(pinoCarro4, HIGH);
        delay(5);
        digitalWrite(pinoCarro4, LOW);
    }

}

//Função para movimentar a cabeça de impressão para a esquerda
void cabecaParaEsquerda(int espacamento){

    for(int counter = 0; counter < 4 * espacamento; counter++){
```

```
        digitalWrite(pinoCarro4, HIGH);
        delay(5);
        digitalWrite(pinoCarro4, LOW);
        digitalWrite(pinoCarro3, HIGH);
        delay(5);
        digitalWrite(pinoCarro3, LOW);
        digitalWrite(pinoCarro2, HIGH);
        delay(5);
        digitalWrite(pinoCarro2, LOW);
        digitalWrite(pinoCarro1, HIGH);
        delay(5);
        digitalWrite(pinoCarro1, LOW);
    }

}

//Desliga as funções das portas
  void desligaFuncoes(){

    digitalWrite(pinoRolete1, LOW);
    digitalWrite(pinoRolete2, LOW);
    digitalWrite(pinoRolete3, LOW);
    digitalWrite(pinoRolete4, LOW);
    digitalWrite(pinoCarro1, LOW);
    digitalWrite(pinoCarro2, LOW);
    digitalWrite(pinoCarro3, LOW);
```

```
    digitalWrite(pinoCarro4, LOW);
    digitalWrite(agulha1, LOW);
    digitalWrite(agulha2, LOW);
    digitalWrite(agulha3, LOW);
    digitalWrite(agulha4, LOW);
    digitalWrite(agulha5, LOW);
    digitalWrite(agulha6, LOW);

}
```

ANEXO A – Classe Java SerialTest de Arduino Playground

```
import java.io.InputStream;
import java.io.OutputStream;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.util.Enumeration;

public class SerialTest implements SerialPortEventListener {
    SerialPort serialPort;
    /** The port we're normally going to use. */
    private static final String PORT_NAMES[] = {
        "/dev/tty.usbserial-A9007UX1", // Mac OS X
        "/dev/ttyUSB0", // Linux
        "COM3", // Windows
    };
    /** Buffered input stream from the port */
    private InputStream input;
    /** The output stream to the port */
    private OutputStream output;
    /** Milliseconds to block while waiting for port open */
    private static final int TIME_OUT = 2000;
    /** Default bits per second for COM port. */
```

```
private static final int DATA_RATE = 9600;

public void initialize() {
    CommPortIdentifier portId = null;
    Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

    // iterate through, looking for the port
    while (portEnum.hasMoreElements()) {
        CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.nextElement();
        for (String portName : PORT_NAMES) {
            if (currPortId.getName().equals(portName)) {
                portId = currPortId;
              break;
            }
        }
    }

    if (portId == null) {
        System.out.println("Could not find COM port.");
        return;
    }

    try {
        // open serial port, and use class name for the appName.
        serialPort = (SerialPort) portId.open(this.getClass().getName(), TIME_OUT);
```

```
        // set port parameters
        serialPort.setSerialPortParams(DATA_RATE,
        SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

        // open the streams
        input = serialPort.getInputStream();
        output = serialPort.getOutputStream();

        // add event listeners
        serialPort.addEventListener(this);
        serialPort.notifyOnDataAvailable(true);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}

/**
 * This should be called when you stop using the port.
 * This will prevent port locking on platforms like Linux.
 */
public synchronized void close() {
    if (serialPort != null) {
        serialPort.removeEventListener();
        serialPort.close();
    }
}
```

```
    /**
     * Handle an event on the serial port. Read the data and print it.
     */
    public synchronized void serialEvent(SerialPortEvent oEvent) {
        if (oEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
            try {
                int available = input.available();
                byte chunk[] = new byte[available];
                input.read(chunk, 0, available);

                // Displayed results are codepage dependent
                System.out.print(new String(chunk));
            } catch (Exception e) {
                System.err.println(e.toString());
            }
        }
        // Ignore all the other eventTypes, but you should consider the other ones.
    }
    public static void main(String[] args) throws Exception {
        SerialTest main = new SerialTest();
        main.initialize();
        System.out.println("Started");
    }
}
```

ANEXO B – Trecho do artigo "Energias renováveis contra o aquecimento global" de autoria do Greenpeace Brasil

Energias renováveis contra o aquecimento global (GREENPEACE, 2010)

A Nasa (agência espacial americana) anunciou no início de 2010 que a década que terminou em 31 de dezembro de 2009 foi a mais quente já registrada desde 1880, ano em que a moderna medição de temperaturas ao redor do planeta começou. A mesma década também teve os dois anos de maior intensidade de calor em mais de um século – 2005, o mais quente do período, e 2009, o segundo mais quente.

O aumento da temperatura do planeta é consequência de ações humanas, especialmente tomadas a partir da Revolução Industrial, no século 18. Ela promoveu um salto tecnológico e o crescimento das civilizações como nunca vistos antes. Impulsionou também uma taxa inédita e perigosa de poluição e degradação da natureza.

A indústria floresceu baseada na queima de carvão e petróleo, duas fontes de energia encontradas na natureza cujo calor movimenta usinas, indústrias e economias gigantescas, como a dos Estados Unidos, da Europa e da China. Acontece que, com a queima, o carvão e o petróleo liberam no ar volumes gigantescos do gás dióxido de carbono (CO2). A exploração sem controle de outra matéria-prima, a florestal, também jogou outros milhões de toneladas desse gás no ar.

O CO2, também chamado de gás carbônico, é parte da atmosfera terrestre e forma uma capa ao redor da Terra, que faz a temperatura do planeta adequada para a manutenção da vida – inclusive a humana. Esse processo se chama efeito estufa. O problema, portanto, não é o mecanismo natural em si, mas a interferência feita pelo homem. O volume extra de CO2 e de outros gases, como o metano, que foram para a atmosfera a partir da Revolução Industrial engrossou a capa, de forma que a temperatura passou a subir perigosamente.

Um planeta mais quente desequilibra o ultra-sensível sistema climático da Terra. Como consequência, o gelo dos polos derrete e eleva o nível médio dos oceanos, ameaçando populações costeiras; tempestades se tornam mais frequentes, intensas e perigosas, assim como

ondas de calor; biomas como a Amazônia são ameaçados pela alteração no sistema de chuvas. Populações já vulneráveis ficam com a corda no pescoço, sofrendo impactos na produção de alimentos, fornecimento de água, moradia.